PUBLICATIONS

G.N. Ramachandran
Mathematical Philosophy Group
Indian Institute of Science
Bangalore 560 012

VOLUME - II   MR 11 - 16

VOLUME - II　　　MR 11 - 16

CONTENTS

PRINCIPLES OF DEDUCTION

(Draft 1 prepared in 1979)

Lecture 8 of the book
"Principles of Practical Logic"
Under preparation

Matphil Reports No. 11 August 1980

P R I N C I P L E S   O F   D E D U C T I O N

(Draft 1 prepared in 1979)

Lecture 8 of the book
"Principles of Practical Logic"
under preparation

Matphil Reports No. 11 August 1980.

Introduction to Lecture 8

This draft was prepared in July 79 when the first draft of the book "Principles of Practical Logic" was written. This was the last lecture that was written in full at that time and it formed the first section of the part of the book dealing with the use of deduction in logic, and the precise significance of the relation of implication. There was an unfortunate break in the writing of the book thereafter for a year.

This draft is partly mathematical, and partly dealing with logic as treated in Indian philosophy. It is pointed out that many relationships that are known now-a-days, e.g. those between implication on the one hand, and the logical operations of conjunction and disjunction on the other, are implicit in the theory of logical deduction (anumana) as given in the old Indian texts. The examples presented in this lecture are, therefore, derived from those given in such texts, of which the famous "hill-smoke-fire" [illegible] is best known. Even in English, there is the saying: "There is no smoke without fire'. This particular sentence, namely one dealing with the fact that we can deduce the existence of fire by observing the occurrence of smoke, is very widely used in Indian logical texts. The various different ways in which this deduction may be expressed and the forms in which an argument can be framed, so as to form a connected and coherent set of statements, are to be found in our learned texts (sastras). In fact, the learning of logic (Nyaya) and the science of disputation (tarka sastra) was essential for all scholars in Ancient India.

In the revision of this chapter, which will be included in the final draft of the book "Principles of Practical Logic", the historical aspects will be given only briefly in order to focus attention on the mathematical theory and logical concepts.

Hence this first draft has been kept bound, so that those who are interested in the close connection between logical deduction in Ancient India and modern Mathematical Logic may have some ideas of these from this account.

LCM 8
Draft 1
21-7-79

Lecture 8. PRINCIPLES OF DEDUCTION

8.1. Review of the structure of logic.

In the previous lectures we had considered how atomic statements (terms) can be connected to one another by means of logical operators and how an argument containing a number of such connected terms can lead to an output, which will also be in the form of elementary atomic statements.

In a way we had seen that all the connective that are involved in classical logic are describable in terms of a very small number of them namely E , N , P and R . Even among these, we can show that R can be described in terms of N and P . Further still, if one uses the so-called "nand" operator, $\not{P}$, then, in combination with E , every other connective can be represented.

These facts are well known in classical logic, particularly as it is applied to computer science. However, in practical logic we find that two more connectives of the type Q and S are essentially required. These occur when two different statements refer to the same final conclusion, and we wish to know which one is operative and how they are connected in producing the outcome. This means that S is absolutely necessary and is part and parcel of the logical machinery which has been used ever since man started producing arguments and discussing things. Analogous to the negation N , we thus have a second negation operator M (which is equivalent to N in classical logic), and in terms of M , Q and S are interrelated in the same way as P and R are related <u>via</u> N . Therefore, we need only S and M as essential additions to P and N . However,

- 3 -

just as the equivalence operator E is essential in the logic of connections between different arguments leading to a new term, another operator A is absolutely necessary as the corresponding " identity " matrix for comparing two statements referring to the same entity and finding the consequence of its superposition, when Q , S and M are introduced.

In other words, we have, so to say, the set E , N , P , R , forming the complex of logical operators (not all independent) which refer to the usual deduction, as discussed in classical logic. At the same time, we also have a new set A , M , Q , S , which are also needed; but which, as we will show later, really refer to a four-state algebra, unlike the two-state algebra for E , N , P and R . Thus, we see that, essentially, everything in logic is capable of

- 4 -

mathematical representation in terms of the above fundamental connectives, and that the ideas that are involved are really quite simple.

8.2. Significance of " if .... then " .

Although we have talked, so far, only of such simple connections, in which the statement is always of the type " a and b is c " or " a or not b is d " etc., in which the conclusion about c or d is uniquely given in terms of the states of the input statements, this need not always be so even if these self-same connectives are used. (We saw some examples in chapter in connection with the binary relations involving P and R). Keeping this in the background, we would like to give an introduction to a new connective, "if". This is a word which is universally used in all arguments in

- 5. -

ordinary language and should therefore be capable of being discussed in simple terms, without any formal logical symbolism, not to talk of mathematical definitions. Therefore in the first few sections of this lecture, we shall give a commonsense description of what is the meaning of a sentence which has the form " if .... then .... " .

It is obvious that, in any argument which contains the derivation of something new from something already known, there always occurs this type of relationship —— namely " if $\underline{\underline{a}}$ is true, then $\underline{\underline{b}}$ is true " The way in which such a conditional statement is used to derive definite and unconditional results is as follows :

Known that
    " if $\underline{\underline{a}}$ is true, then $\underline{\underline{b}}$ is true " .
and given that
    " the statement $\underline{\underline{a}}$ is true "
it follows that
    " the conclusion $\underline{\underline{b}}$ is true " .

(8.1)

- 6 -

The essence of this is that we first have a general statement (the use and significance of the word general will become clear as we go further on). In this general statement, we observe that " $\underline{\underline{a}}$ is true " implies " $\underline{\underline{b}}$ is true " . Here again, the word implies is used as it is understood in ordinary language ; its precise mathematical definition will be given later. What we mean is that, if the former statement is valid, then the latter statement becomes valid. Straight-away, we may mention that only this matters, and we do not at all care about what is the significance of the antecedent (first statement) being false. Hence the conditional implication, " if .... then .... " has really only one half the content as all the other types of connectives such as E , P , R etc., which have discussed so far. Thus, we only relate a_T with b_T and say that a_T always corresponds to b_T .

- 7 -

We also say that this correspondence is in the direction from $\underline{\underline{a}}$ to $\underline{\underline{b}}$. Further, a_F may correspond to anything, and gives no definite information about $\underline{\underline{b}}$. These are not rigorous definitions of the property of implication, but rather an explanation of what one normally means by saying " if $\underline{\underline{a}}$, then $\underline{\underline{b}}$ " . It really means that, if we know from some source that $\underline{\underline{a}}$ is true, then we can conclude that $\underline{\underline{b}}$ is true, if we are also aware of the fact that " if $\underline{\underline{a}}$, then $\underline{\underline{b}}$ " is always valid.

The best way of understanding the principle behind this type of logical deduction is really to go back to the origin of the science of logic, which apparently took place in the first millenium B.C. There are the great books of Aristotle, in Greece, and correspondingly we have the volumes on Nyaya

- 8 -

philosophy in India. Aristotle was very much more ctitical and analytical than the proponents of the Nyaya system, and he produced a great system dealing with the types of logical statements and the logical interconnections between tham, which can all be distinguished and described using a suitable grammar for logic. On the other hand, the Nyaya philosophers were bothered more with the essence of the methods of obtaining knowledge. They did not think of logic as a mere system of grammar. They considered logic as an intimate part of the science of epistemology. To a small extent,we are also being following this method of approach in our description of logic in these lectures. In other words, we do not say that the statement " $\underline{\underline{a}}$ and $\underline{\underline{b}}$ is true " is equivalent to " $\underline{\underline{c}}$ is true " means that it has relevance only to the abstract situation governing the interrelation between $\underline{\underline{a}}$, $\underline{\underline{b}}$ and $\underline{\underline{c}}$, and their truth values. Instead, we say that the best way of finding out the information about $\underline{\underline{c}}$ when it is connected by such an equivalence to the statement " $\underline{\underline{a}}$ and $\underline{\underline{b}}$ is true " is by examining whether each of $\underline{\underline{a}}$ and $\underline{\underline{b}}$ are, in fact, true or false. According to the information that we thus obtain, we can deduce about the validity of the statement $\underline{\underline{c}}$, and utilise this information thereafter.

I hope I have made clear the distinction between formulating a purely abstract, systematic, rigorous, formulation in any science, and the development of methods for applying these rules to obtain practical results in that field. If one may take an example, the difference is the same as what exists between the mathematics of pure

- 10 -

mathematicians and the applied mathematics of theoretical physicists who use it for getting results in physical applications. Pure mathematics consists of theorems, and these theorems always have the form of a logically self-consistent set of statements, which denote the interconnection between the assumptions and the conclusion of a theorem. In fact, mathematicians even go so far as to say that they do not at all care whether the assumptions correspond to reality, or whether they are "true" in some sense. All they are interested in is the fact that if the assumptions are true, then the conclusion is true. On the other hand, in the applications of mathematics, particularly for theories in physics, we are interested in actually working out the nature of the conclusion, and in finding out what it can be used for, if we are given that the premises,

- 11 -

or the starting assumptions, are known to be true. In such a theory, mathematics is only a tool, for shortening the logical deductions that are needed to go from the beginning to the end. The theory has a much wider range of existence than what is abstracted in the mathematical interrelationship that is utilized in the derivation of the conclusion.

If we may take another analogy, we know that a language is much more than its grammar. In fact, the best grammarians are often the worst poets. The writer who uses a language to express his ideas must, of course, obey the rules of grammar. But very often, he knows these rules by intuition rather than by a rigorous study spread over many years. The grammarian for language, is the equivalent of the mathematical logician for logic and the pure mathematician

- 12 -

for physical theory. The author, or the poet, who produces a piece, is the equivalent of the physicist in mathematical physics and similarly, the thinker who uses intuition for obtaining knowledge about anything, rather than the strict rules of pure logic, is the "poet" of logic.

Having made this diversion, we will now try to arrive at the precision required for the description and utilisation of the operation of implication <u>via</u> the arduous path that the scholars have pursued over the ages. I have always found that the best way of introducing a subject to the non-initiates is to sequentially follow the same path as the one by which discoveries were made during the ages in that subject. Therefore, we will spend some time in discussing the approaches made by the philosophers of ancient India, particularly in the Nyaya system of logic. We will indicate inter alia how great an emphasis they have put on the actual validity of the assumptions from which deductions are made.

8.2. The nature of deduction.

As already mentioned, the application of methods of deduction, or inference, is as ~~old~~ as man. We may quote as interesting statement summarising the scope and application of inference according to the Nyaya system:

" Inference operates neither with regard to things not known, nor with regard to those known definitely for certain ; it functions only with regard to things that are doubtful " . (8

In other words, inference, which is made from something known to obtain knowledge regarding something unknown, effectively removes the doubt about the unknown. Taking

- 14

the above sentence (8.2), it is clear that if we do not know anything to start with, nothing can be concluded about anything else as a consequence. Therefore inference requires a connection between a first thing ($\underline{\underline{a}}$) that is known, and a second thing ($\underline{\underline{b}}$) that is unknown ; but this connection must also be known to be always existent. Further, as regards the second thing, there should not be any certainty, i.e. either that it is correct or that it is wrong, in our knowledge of it. If either of these is so, then there is no need at all for the application of inference, to know about $\underline{\underline{b}}$, because we already have the full information about it. Therefore, at the starting point of our attempt at obtaining knowledge about $\underline{\underline{b}}$, we have this statement (which we want to investigate) in the doubtful state, namely b_D . This state b_D is converted either to b_T , or b_F , as a result of the inference.

For convenience, we shall take this deduced state of $\underline{\underline{b}}$

- 15 -

to be the state of truth. (This leads to no loss of generality, because if it is the falsity of $\underline{\underline{b}}$ that we wish to establish, we can always define a new $\underline{\underline{b}}$ which is equal to the negation of the old $\underline{\underline{b}}$, and say that we wish to prove the new $\underline{\underline{b}}$ to be true.) This is proved from two items of "knowledge" , or "information" :

(i) There exists a suitable relation between the truth of $\underline{\underline{a}}$ and the truth of the statement to be studies, namely $\underline{\underline{b}}$.

(ii) The statement $\underline{\underline{a}}$ itself is known to be true. (8.3)

In the second part (ii) also, the affirmative form of the statement " $\underline{\underline{a}}$ is <u>true</u> " is again not an orbitary choice, because if the information is that $\underline{\underline{a}}$ is false, the old statement $\underline{\underline{a}}$ can be substituted by the negation of it to obtain a new statement, which has the form, " $\underline{\underline{a}}$ is true" .

Thus, (8.3) encompasses not only the most general, but probably the only form, of the assumptions required to make a deduction, or inference, from the known regarding the unknown. In this connection, we may remind ourselves that, when we considered the connectives P and R, we first defined the nature of the relation $\underline{\underline{a}}$ P $\underline{\underline{b}}$ and $\underline{\underline{a}}$ R $\underline{\underline{b}}$, where $\underline{\underline{a}}$ and $\underline{\underline{b}}$ are both in the affirmative form. Then, using the negation operator N suitably, we showed that this could always be converted to a relation involving P and R , such as between $\neg\underline{\underline{a}}$ and $\underline{\underline{b}}$, $\underline{\underline{a}}$ and $\neg\underline{\underline{b}}$, and $\neg\underline{\underline{a}}$ and $\neg\underline{\underline{b}}$. This is a very general technique that can be adopted to simplify the study of the essential nature of any connective, and we shall employ it also for our study of the new connective of implication involved in the process of deduction. Thus, we need spend all our time only to understand clearly what is meant by " $\underline{\underline{a}}$ implies $\underline{\underline{b}}$ " , under the conditions stated in (8.3).

- 17 -

Before we go to the application of (8.3), we shall again make one more diversion, and look at the way in which an inferential set of statements are put down in the Nyaya system. It consists of the following five members (examples are also given).

(i)	<u>Pratijna</u>	(proposition)	the hill is on fire	(8.
(ii)	<u>Hetu</u>	(Reason) :	because it smokes	
(iii)	<u>Udaharana</u>	(Example) :	whatever emits smoke has fire	
(iv)	<u>Upanaya</u>	(Application) .	so is the hill	
(v)	<u>Nigamana</u>	(Conclusion) :	therefore the hill is on fire	

The line of argument is pretty clear. If we wish to put it in a brief way, we may say as follows.

" We conclude that the hill is on fire, because it smokes, since we already know that whatever emits smoke has fire " . (8.5)

If we wish to put it in the form of the modern logical apparatus and if we use the symbol $\Longrightarrow$ for the operation of implication ($\underline{\underline{a}} \Longrightarrow \underline{\underline{b}}$ is equivalent to " if $\underline{\underline{a}}$, then $\underline{\underline{b}}$ "), we will put the above argument in terms of three steps,

$$\left.\begin{array}{llll} & \text{(i)} & \text{smoke} \Longrightarrow \text{fire} \\ & \text{(ii)} & \text{hill} \equiv \text{smoke} \\ \text{Hence,} & \text{(iii)} & \text{hill} \equiv \text{fire} \end{array}\right\} \qquad (8.6)$$

which correspond, respectively, to (iii), (ii), and (i) in (8.4). (In fact, the later Nyaya logicians have stated that the fourth and fifth contentions in (8.4) are not essential, as was implied in the summarised form (8.5) of (8.4)).

Analysing this further, what we first say is that 8.6 (i) is universally true, i.e. whatever emits smoke has fire, and therefore we assume the truth of the general statement "smoke implies fire" . In the particular case, we equate hill with smoke (statement (ii) in (8.6)). Strictly, the equivalence operation ($\equiv$) which we have put in 8.6 (ii) is not essential, as we will see later, but, to make the analysis simpler, we have used the equivalences in 8.6 (ii) and (iii). All that we have to do is to apply the general properties "smoke" ($\underline{\underline{a}}$) and "fire" ($\underline{\underline{b}}$) , which are connected by 8.6 (i), to the particular case in which "smoke" is associated with "hill" ($\underline{\underline{a}}'$) and "fire" is associated with "hill" ($\underline{\underline{b}}'$) . Then we make a specialization of the general law $\underline{\underline{a}} \Rightarrow \underline{\underline{b}}$ to the form $\underline{\underline{a}}' \Rightarrow \underline{\underline{b}}'$. In this form, we apply the sequence of steps indicate in (8.1) to deduce that $\underline{\underline{b}}'$ is true from the fact that $\underline{\underline{a}}'$ is true.

8.3. The Greek approach to inference and deduction.

We had seen earlier that the question of deducing a consequence from facts, or information, already available, was considered from the philosophical point of view both in Greece

- 20 -

and in India. In the last section, we had briefly mentioned the Indian Nyaya approach in which the idea of deduction is essentially used to infer, or deduce, new information, or fact, from previously known data, or facts. On the other hand, the approach in ancient Greece, typified by the classical work of Aristotle, was more formal and abstract ; and, in a sense more akin to modern symbolic logic. The word "syllogism" was used for the unit that was used in any argument. In essence, the syllogism of Aristotle is very similar to the syllogism of the Nyaya philosophers, in that it could be defined as "a discourse, in which, certain things posited, something else necessarily follows from their being so" . This is a description of the ancient Greek syllogism. We may illustrate it by a simple example of the mood Barbara, the first of the various moods that have been written down in

- 21 -

the Aristotelian system. A translation of Aristotle's description of this mood Barbara (the name Barbara is due to a much later logician Peter of Spain) is as follows :

" For if A (is predicated) of all B , and B of all C , it is necessary for A to be predicated of all C " . (8.7)

An example of this is the following :

(i) All men are mortal.
(ii) All Greeks are men.
Therefore,
(iii) All Greeks are mortal. (8.8)

The reasonableness of the final deduction in the third line of this syllogism, from the first and second lines, is obvious and requires no comment. However, the similarity of these to the three steps in the Nyaya syllogism is not exact, although superficially, the two appear to be alike. If,

- 22 -

however, we rearrange (8.8) in another form, as given below in (8.9), the similarity to the Nyaya syllogism becomes almost exact.

(i) All men are mortal.
(ii) A Greek is a man. (Socrates is a man).
Therefore,
(iii) A Greek is mortal (Socrates is mortal). (8.9)

The essential difference between (8.8) and (8.9) is that, in (8.8) all the three statements in the syllogism refer to all members of a class —— in (i) to all men, in (ii) to all Greeks and again in (iii) to all Greeks, while in (8.9), we apply the generality expressed in (i) for all men to the particular case in (ii) of a Greek (Socrates), who is a man, so that we obtain factual knowledge in (iii) about the latter. Suppose we now remove the "all" in each of the three

- 23 -

statements in (8.8). We then get the following :

(i)	Man is mortal.	(8.10)
(ii)	Greek is man.	
(iii)	Greek is mortal.	

This is yet another form of the syllogism in (8.8) which is different from the Aristotelian one, namely (8.8), and also the Nyaya one, namely (8.9); but yet, it is completely valid in pure logic, the deduction in the third statement duly following from the truth of the statements (i) and (ii). In fact, in our initial discussion of the logical basis of deduction, we shall use this last form (8.10), because it completely fits in with the so-called "propositional logic", which is what we have been discussing so far in these lectures.

Yet it is rather interesting that the early logicians preferred the forms (8.8) and (8.9). In fact, (8.8) and

- 24 -

(8.10)are practically equivalent in their information content ; but their logical forms are different, in that (8.10) is within the framework of the system we have developed so far, while (8.8) requires what is known as a quantifier i.e, an adjective qualifying the subject, such as "all" , "some" , or "none". This is a more complicated form of logic, called "predicate calculus" and we shall discuss this only much later. Still it is interesting to see that some of the earliest analyses of logical arguments were of the forms belonging to predicate calculus.

On the other hand, the Nyaya syllogism which is given in (8.4) and (8.5) regarding "hill" and "fire" , and the example given in (8.9) regarding "Greek" and "mortal" are ~~are~~/~~also~~ really of the type used in propositional calculus.

The only speciality is that the first statement is a general statement, and the second and third are particular statements. For instance, in (8.9) we first say that <u>all</u> men are mortal but the second and third are of the form " Socrates is a man" and "Socrates is mortal" . To get the last deduction, the first statement, "<u>all</u> men are mortal " is taken, in its particular form, to mean that " every man is mortal ", or "a man is mortal" . We shall not pursue this further, but it is important to distinguish between a formal syllogism of the type (8.10), which is syntactically true in propositional calculus, and the form (8.9), which, though it is practically equivalent to (8.10), is very often the one that is used to deduce the factual conclusion (iii). It requires for this purpose, the semantic content of the content of the conditional general statement (i) —— (compare also the case of the hill-smoke-fire example, whose abstract logic, given in (8.6), takes a real shape only when the argument giving the general proposition (iii) of (8.4) is used).

In fact, it is rather surprising that the later Greek logicians, such as Zeno of the Stoic school, in the early centuries after Christ, have described examples of syllogisms akin to the abstract form of the type (8.10). According to the Stoics, there are five basic schemas of logical arguments having a deduction in it, which are given below in Table 8.1. In each of these, we give the verbal statement first and follow it by the symbolic logical statement on the same line.

Table 8.1. Five basic schemas of Stoic logicians.

Name as per our system	Description	Symbolism
Implication	(1) If the first, then the second; The first ; Therefore, the second.	$\underline{\underline{a}} \Rightarrow \underline{\underline{b}}$ $\underline{\underline{a}}$ $\underline{\underline{b}}$
Reverse implication	(2) If the first, then the second; Not the second ; Therefore, not the first.	$\underline{\underline{a}} \Rightarrow \underline{\underline{b}}$ $\neg \underline{\underline{b}}$ $\neg \underline{\underline{a}}$
Nand	(3) Not both first and the second; The first ; Therefore, not the second.	$\neg(\underline{\underline{a}} \cdot \underline{\underline{b}})$ $\underline{\underline{a}}$ $\neg \underline{\underline{b}}$
Exclusive or	(4) The first or the second ; The first ; Therefore, not the second.	$\underline{\underline{a}}\ \underline{\vee}\ \underline{\underline{b}}$ $\underline{\underline{a}}$ $\neg \underline{\underline{b}}$
Exclusive or	(5) The first or the second; Not the first ; Therefore, the second.	$\underline{\underline{a}}\ \underline{\vee}\ \underline{\underline{b}}$ $\neg \underline{\underline{a}}$ $\underline{\underline{b}}$

- 28 -

These are beautifully simple examples of the consequences of a conditional statement connecting two components in the first line, followed by a definite informational statement in the second about one of the components, which lead to the deduction of a definite statement, (yes, or no), about the other component of the pair. In fact, if we look at these carefully, we find that a number of logical operators of the type we have already studied are all involved. In (1) and (2) of Table 8.1, we have the connectives "implication" and "reverse implication", which we shall discuss in this and the next lecture (when we will show that they are equivalent to one type of "or" (R) operation); (3) corresponds to "nand", and (4) and (5) both have the "exclusive or" as the connective involved. It is indeed interesting that all these connectives had caught the fancy.

- 29 -

of the Greek toics two thousand years ago.

Books of logic at an elementary level, which go also into the early history of the subject, discuss several other possible combinations of statements in a syllogism discussed in the calssical systems. However, we shall show that all of them, irrespective of the nature of the argument, are all derivable from just one, namely what we have given in (8.1), and what we last discussed in (8.10) —— namely the Nyaya syllogism, **or** the mood <u>Barbara</u> of Aristotle, reduced to the logical form of propositional calculus. In fact, what we shall show is that "deduction" forms one vital, fundamental, proces in every step of a logically connected argument. There is only one principle involved in this. If this is defined and understood, and, if in addition, we clearly understand what is

- 30 -

meant by "not", all the other forms can be related to this single one by suitable combinations of these two connectives.

8.4. Syntax and semantics of inference.

In the previous two sections 8.2 and 8.3, we have discussed briefly the approach to deduction, or inference, as taken by the ancient Indian philosophers and the Greek philosophers. **Perhaps** it may be worthwhile mentioning some of the essential differences between the two. Quoting from the book "Outlines of Indian Philosophy" , by M.Hiriyanna, we may mention the following :

"... the verbal view of logic which is common in the West is rejected here. It was never forgotten in India that the subject-matter of logic is thought, and not, in any sense, the linguistic forms in which it may find expression".

- 31 -

In fact, the celebrated Italian philosopher, Croce, says the following about Indian logic in his book " Logic " :

> "Indian Logic studies the naturalistic syllogism in _itself_ as internal thought, distinguishing it from the syllogism for _others_. . . . . It does not make the verbal distinctions of subject, copula and predicate; it does not admit classes of categorical and hypothetical, of affirmative and of negative judgments. All these are extraneous to Logic, whose object is the constant: knowledge considered in itself".

In spite of this, it is interesting that the Indian logicians had extremely clear ideas of when an inference is valid, and exactly under what conditions it cannot be arrived at. In fact the purpose of logic,according to the Nyaya school, and others associated with it, was to categorise "patterns of debate" . The very word Nyaya itself means "argument" . In this connection, the opening argument found in the Buddist text _Kathavattu_ (in the early centuries B.C.),

- 32 -

is worth noting :

"If my first statement ($\underline{\underline{a}}$) was true, then you should consent to my second ($\underline{\underline{b}}$) . You are wrong to say that $\underline{\underline{a}}$ is true and $\underline{\underline{b}}$ is not true. If $\underline{\underline{b}}$ is not true, then $\underline{\underline{a}}$ is not true. You are wrong to say that $\underline{\underline{a}}$ is true, but $\underline{\underline{b}}$ is not true" . (8.11)

These sentences smack of the rigour of mathematical logic of the 20th century. In fact, if the four statements contained therein are written symbolically in the notation adopted in our lectures, they take the following forms :

$$\underset{\text{(i)}}{(\underline{\underline{a}} \Rightarrow \underline{\underline{b}})} \equiv \underset{\text{(ii)}}{\neg(\underline{\underline{a}} \circ \neg\underline{\underline{b}})} \equiv \underset{\text{(iii)}}{(\neg\underline{\underline{b}} \Rightarrow \neg\underline{\underline{a}})}$$
$$\equiv \underset{\text{(iv)}}{(\underline{\underline{a}} \not\Rightarrow \neg\underline{\underline{b}})} \qquad (8.12)$$

It is indeed surprising that the equivalence of these four different ways of expressing the same logical fact, namely

- 33 -

" $\underline{\underline{a}}$ implies $\underline{\underline{b}}$ ", should have been written so clearly in this logical work produced more than 20 centuries ago.

8.5. Truth table for implication.

Therefore, we shall start our understanding of implication by examining these four statements, and trying to understand, in a common-sense way, why they do not mean different things; but that they all really mean exactly one thing **— namely the** relation $\underline{\underline{a}} \Longrightarrow \underline{\underline{b}}$. For this purpose, we have to go back to section 8.2, and look into the statement given in (8.1). The real meaning of the implication $\underline{\underline{a}} \Longrightarrow \underline{\underline{b}}$ is that, if the fact that $\underline{\underline{a}}$ is true is valid, then $\underline{\underline{b}}$ **is also a valid statement.** is **It is, in fact a** proposition, in the sense that it is always true. Given that this general relationship between $\underline{\underline{a}}$ and $\underline{\underline{b}}$ holds, namely that "$\underline{\underline{a}}$ implies $\underline{\underline{b}}$", we now ask as to **what** can be said about

$\underline{\underline{b}}$, given something about $\underline{\underline{a}}$. The answer to this is already contained in (8.1) — namely that, if the asserted statement $\underline{\underline{a}}$ is actually true, then necessarily, we get the conclusion that $\underline{\underline{b}}$ is true. Therefore, even at the risk of repeating the same, we restate (8.1) as follows :

$$\underline{\underline{a}} \Longrightarrow \underline{\underline{b}} \text{ means } a_T \longmapsto b_T \qquad (8.13)$$

We can now ask the question: Suppose $\underline{\underline{a}}$ were actually not true, and the starting point is the term a_F; what can we say about $\underline{\underline{b}}$? (The pxoper form of the answer to this is the one that decides the range, importance, significance, and the types of applications that are possible for the implication connective). The answer, actually, is quite simple; if $\underline{\underline{a}}$ were false, then the implication, $\underline{\underline{a}} \Rightarrow \underline{\underline{b}}$, says nothing definite about <u>anything</u>. In other words, the " if .... then " connection

- 35 -

connects only the truth value T of $\underline{\underline{a}}$ with the truth value T of $\underline{\underline{b}}$. If $\underline{\underline{a}}$ were a_F, then nothing whatever follows about $\underline{\underline{b}}$. This second result is usually given in standard logic books in the following form :

$$\left.\begin{array}{c} \underline{\underline{a}} \Longrightarrow \underline{\underline{b}} \text{ means that} \\ a_F \longmapsto b_T \text{ and } a_F \longmapsto b_F \text{ are both true.} \end{array}\right\} \quad (8.14)$$

Consequently, a_F could lead to both b_T and b_F, and either possibility is not forbidden, so that a_F can lead to no useful result about $\underline{\underline{b}}$. This means that $\underline{\underline{b}}$ is in the doubtful state b_D which we have introduced **in our lectures.** Hence, we modify (8.14) to read as follows :

$$\underline{\underline{a}} \Longrightarrow \underline{\underline{b}} \quad \text{means} \quad a_F \longmapsto b_D \qquad (8.15)$$

Thus, (8.13) and (8.15) together give the full consequences

- 36 -

of assuming the existence of the relation $\underline{a} \Rightarrow \underline{b}$, for the two possibilities, a_T or a_F , of $\underline{a}$. From these purely verbal statements contained in (8.13) and (8.15) we shall work out their consequences.

of assuming the existence of the relation $\underline{\underline{a}} \Longrightarrow \underline{\underline{b}}$, for the two possibilities, a_T or a_F, of $\underline{\underline{a}}$. ~~We shall not try to put this in the matrix form right now, but reserve it for the next lecture~~. ~~Here,~~ From these purely verbal statements contained in (8.13) and (8.15), we shall work out their consequences.

What we find, in effect, is that, under the implication $\underline{\underline{a}} \Longrightarrow \underline{\underline{b}}$, which we have assumed to exist, a_T corresponds <u>only</u> to b_T. Therefore a_T <u>will never lead</u> to the consequence b_F. Symbolically this can be written as follows :

$$\underline{\underline{a}} \Longrightarrow \underline{\underline{b}} \quad \text{means} \quad a_T \not\longmapsto b_F \qquad (8.16)$$

This result is a straightforward logical consequence of (8.13) and (8.14), obtained from commonsense, and it is the equivalent of (iv) of (8.12) which symbolises the fourth statement of the Kathavattu argument quoted in (8.11).

If we combine the data in (8.13), (8.14) and (8.16), we obtain completely all that are needed to construct the truth table given below in Table 8.2 for " $\underline{\underline{a}}$ implies $\underline{\underline{b}}$ " .

Table 8.2. Truth table for the . . relation $\underline{\underline{a}} \Longrightarrow \underline{\underline{b}}$.

(a) Binary

$\underline{\underline{a}}$	$\underline{\underline{b}}$
T	T
F	D
D	D

(b) Ternary

$\underline{\underline{a}}$ \ $\underline{\underline{b}}$	T	F
T	T	F
F	T	T

8.6. Interpretations of the implication relation $\underline{\underline{a}} \Longrightarrow \underline{\underline{b}}$ in the forward and backward directions.

The truth table 8.2 for the ternary relation

- 38 -

$$(\underline{\underline{a}} \Longrightarrow \underline{\underline{b}}) = \underline{\underline{c}} \qquad (8.17)$$

is the one that is commonly employed in logic texts as the definition giving the properties of this logical connective. We shall also follow this conventional approach and work out its consequences, purely from the data exhibited in this truth table. However, we shall only use commonsense, and not the matrix method, in this lecture. We shall focus our attention mainly on the four different equivalent expressions of (8.17) given in the four parts of the Kathavattu argument and symbolized in (8.12).

Considering first $\underline{\underline{a}} \Longrightarrow \underline{\underline{b}}$ **itself,** we clearly see from Table 8.2 that the truth of $\underline{\underline{a}}$ corresponds to the truth of $\underline{\underline{b}}$. Also, as already mentioned, when $\underline{\underline{a}}$ is false, both the possibilitites and F are possible for $\underline{\underline{b}}$ and therefore nothing comes out positively about $\underline{\underline{b}}$. On the other hand, if $\underline{\underline{a}}$ is true, $\underline{\underline{b}}$ can never be false. The last aspect is extremely important, because it is this part of the full truth table 8.2(b), which is the most significant one for obtaining the transformation of (8.17) into the other equivalent forms in (8.12). Putting all these together, we have the following correspondences for the logical relation " $\underline{\underline{a}}$ implies $\underline{\underline{b}}$"

$$\underline{\underline{a}} \Longrightarrow \underline{\underline{b}} \quad \text{means that}$$

$$a_T \longmapsto b_T \; ; \; a_T \not\longmapsto b_F \qquad (a)$$

$$\text{and} \quad a_F \longmapsto b_T \; ; \; a_F \longmapsto b_F \; ; \; \text{or} \; a_F \longmapsto b_T \qquad (b) \qquad (8.18)$$

Now, the terms $\underline{\underline{a}}$ and $\underline{\underline{b}}$ can be interchanged, so that we can work out the consequences for $\underline{\underline{a}}$ corresponding to the T and F states of $\underline{\underline{b}}$. On making this re-interpretation, we get from Table 8.2(b) the following :

$\underline{\underline{a}} \Longrightarrow \underline{\underline{b}}$ also means that

$$b_F \longmapsto a_F \;;\; b_F \longmapsto a_T \qquad \text{(a)}$$
$$b_T \longmapsto a_T \;;\; b_T \longmapsto a_F \;;\; \text{or } b_T \longmapsto a_D \qquad \text{(b)} \qquad (8.19)$$

If we compare the two lines (a), (b) of (8.18) with the corresponding ones in (8.19), it is readily seen that a_T and b_F have interchanged their roles (and therefore a_F and b_T have also interchanged their roles). Thus, when $\underline{\underline{b}}$ is false, $\underline{\underline{a}}$ is false, and when $\underline{\underline{b}}$ is true, nothing can be said about $\underline{\underline{a}}$. Also, the correspondence $a_T \not\longmapsto b_F$, takes, in the reverse direction from $\underline{\underline{b}}$ to $\underline{\underline{a}}$, the form $b_F \not\longmapsto a_T$ Using these interrelationships between the sets of correspondences in (8.18) and (8.19), we can therefore say that

$$(\underline{\underline{a}} \Longrightarrow \underline{\underline{b}}) \text{ is equivalent to } (\neg\,\underline{\underline{b}} \Longrightarrow \neg\,\underline{\underline{a}}) \qquad (8.20)$$

This is one of those results which looks peculiar at first glance (In fact, we may ask the question : is $(\underline{\underline{a}} \Longrightarrow \underline{\underline{b}})$ equivalent to $(\underline{\underline{b}} \Longrightarrow \underline{\underline{a}})$? The answer is no — see below).

but ~~By~~ a careful and complete analysis made using the truth table 8.2, (although carried out from first principles) shows that the equivalent form in the reverse sense is really $\neg \underline{\underline{b}} \Longrightarrow \neg \underline{\underline{a}}$. This corresponds to the third of the statements mentioned in the <u>Kathavattu</u> argument (8.11).

It may be worthwhile pondering over this, <u>vis-a-vis</u> the "hill-smoke-fire" example. We say first that "smoke implies fire" ; but then we find from our analysis that it is completely equivalent to saying that "no fire means there will be no smoke" (not "fire implies smoke"). This reverse implication becomes obvious on reflection, even without the detailed analysis. In fact, the Indian texts give a counterexample to show that the statement within the bracket given above need not be true — e.g. a ball of iron can be made red hot as a fire, but it produces no smoke ; on the other hand, absence of smoke always implies sence of fire.

The equivalent forms in (8.20) are absolutely vital for mathematical proofs. Thus $\underline{\underline{a}} \Longrightarrow \underline{\underline{b}}$ can be put in words in a number of ways :

$$\underline{\underline{a}} \Longrightarrow \underline{\underline{b}}$$

(a) Statement $\underline{\underline{a}}$ *implies* statement $\underline{\underline{b}}$
(b) Truth of $\underline{\underline{a}}$ is *necessary* for the truth of $\underline{\underline{b}}$ (
(c) *If* $\underline{\underline{a}}$ is true, *then* $\underline{\underline{b}}$ is true

On the other hand, the statement $\underline{\underline{b}} \Longrightarrow \underline{\underline{a}}$, which is also possible, is quite different from this. Its corresponding verbal interpretations are as follows :

$$\underline{\underline{b}} \Longrightarrow \underline{\underline{a}}$$

(a) Statement $\underline{\underline{a}}$ *is implied by statement* $\underline{\underline{b}}$
(b) Truth of $\underline{\underline{a}}$ is *sufficient* for the truth of $\underline{\underline{b}}$ (8
(c) *If* $\underline{\underline{a}}$ is *not* true, *then* $\underline{\underline{b}}$ is *not* true

The last one (c) of (8.22) uses the equivalence (8.20) applied to $\underline{\underline{b}} \Longrightarrow \underline{\underline{a}}$ and this is what is meant by the "necessary"

- 43 -

condition in (8.22 b). If we use it for $\underline{\underline{a}} \Longrightarrow \underline{\underline{b}}$ itself in (8.21), we then obtain the following three other verbal statements for $\underline{\underline{a}} \Longrightarrow \underline{\underline{b}}$:

$$(\underline{\underline{a}} \Longrightarrow \underline{\underline{b}}) \equiv (\neg\underline{\underline{b}} \Longrightarrow \neg\underline{\underline{a}})$$

(a) Statement $\underline{\underline{b}}$ <u>is implied by</u> statement $\underline{\underline{a}}$

(b) Falsity of $\underline{\underline{b}}$ is <u>sufficient</u> for the falsity of $\underline{\underline{a}}$ (8.23)

(c) <u>If</u> $\underline{\underline{b}}$ is not true <u>then</u> $\underline{\underline{a}}$ is not true

It follows from (8.20) and (8.23) that an implication relation in the forward direction can always be converted into another implication relation in the backward direction, so that, under suitable conditions, an argument can be traced backwards. Thus taking the example "man implies mortal" for $(\underline{a} \Longrightarrow \underline{b})$, we obtain the direct consequence "X is man, hence X is mortal"; or using the reverse form $(\neg\underline{b} \Longrightarrow \neg\underline{a})$, "Y is not mortal, hence Y is not man." However, if X is mortal, <u>it does not follow</u> that X is man. It is very

- 44 -

important to realize that $(\underline{\underline{a}} \Longrightarrow \underline{\underline{b}})$ and $(\underline{\underline{b}} \Longrightarrow \underline{\underline{a}})$ are absolutely independent relations, and either can be true without the other being true. On the other hand, $(\underline{\underline{a}} \Longrightarrow \underline{\underline{b}})$ and $(\neg \underline{\underline{b}} \Longrightarrow \neg \underline{\underline{a}})$ are not only not independent, but are exactly equivalent, and mean the same thing. The best way of convincing ourselves about this without a detailed analysis of the truth table is by expressing them both in terms of a common form equally related to both. This is exactly what is done in the <u>Kathavattu</u> argument. The second part (8.12(ii)) has completely similar relations to the first part and the third part ((i) and (ii) of (8.12)) on either side of it, and hence proves the equivalence of (i) and (iii), which we have derived otherwise in (8.20). To see the principle of this, we make a small diversion to modern logic.

<u>8.7. Relations between implication, conjunction and disjunction</u>.

Let us examine the equivalence which is very commonly used

- 45 -

in logic texts for the representation of an implicational relation via a relation of conjunction ("or"), namely

$$(\underline{\underline{a}} \Longrightarrow \underline{\underline{b}}) \equiv (\neg \underline{\underline{a}} \vee \underline{\underline{b}}) \qquad (8.25)$$

The second expression is readily verified to represent the correspondences in (8.18), for we know that $\underline{\underline{x}} \vee \underline{\underline{y}}$ is false only if both $\underline{\underline{x}}$ and $\underline{\underline{y}}$ are false. Hence, the right hand side of (8.25) is false only if $\underline{\underline{a}}$ is true and $\underline{\underline{b}}$ is false, exactly as in Table 8.2(b). (We shall give examples to illustrate this a little later in this section). Now, we know that an "or" relation between two statements is also expressible in terms of the negation of an "and" relation connecting the negations of the two statements, namely the so-called de Morgan Law (Equation 7.24). If we now apply this to the right hand side of (8.25), we obtain that $(\neg \underline{\underline{a}} \vee \underline{\underline{b}}) = \neg (\underline{\underline{a}} \, . \neg \underline{\underline{b}})$, so that the implication $\underline{a} \Longrightarrow \underline{b}$ is expressible in one more equivalent form containing a "nand" , namely :

$$(\underline{\underline{a}} \Longrightarrow \underline{\underline{b}}) \equiv \neg(\underline{\underline{a}} \cdot \neg\underline{\underline{b}}) \qquad (8.26)$$

This form, given on the right hand side of (8.26), is nothing but the second statement of the <u>Kathavattu</u> argument, namely "you are wrong to say that $\underline{\underline{a}}$ is true and $\underline{\underline{b}}$ is not true". Although we arrived at (8.26) via de Morgan's law, it is also possible to be convinced of its validity straightaway from the truth table 8.2, which gives F only for the combination a_T <u>and</u> b_F, as required by (8.26).

The validity of the right hand sides of both (8.25) and (8.26) as equivalent forms of $\underline{\underline{a}} \Longrightarrow \underline{\underline{b}}$ is very clearly seen using the example of "smoke ($\underline{\underline{a}}$) implies fire ($\underline{\underline{b}}$)". Considering first (8.26), it leads to the statement, "it is not possible that there is smoke and no fire", a form which is there in all ancient arguments. However, the form corresponding to (8.25), namely "not smoke, or fire, must be true" (with "or" as

- 47 -

inclusive or) is not found either in the Kathavatthu argument, or in the five Stoic schemas listed in Table 8.1. The reason for this is quite simple. The idea of "inclusive or" is one of relatively recent origin, and it is intimately connected with the idea of union of sets. Only the "exclusive or" (either, or) was used by the ancients, as in (4) and (5) of Table 8.1. Yet the above application of (8.25) to the "hill-smoke-fire" example becomes quite clear, once we appreciate the significance of the inclusive "or" . Thus, every true possibility comprehended by $\underline{\underline{a}} \Longrightarrow \underline{\underline{b}}$, is covered by taking either the situations in which there is no smoke ($\neg \underline{\underline{a}}$), or those in which there is fire ($\underline{\underline{b}}$) . Thus, the two sentences in (8.25), namely ($\underline{\underline{a}} \Longrightarrow \underline{\underline{b}}$) and ($\neg \underline{\underline{a}} \vee \underline{\underline{b}}$) , are equivalent.

It is indeed strange that, in formal symbolic logic, and in our matrix formalism which we will discuss in the

- 48 -

next lecture, it is this form $(\neg \underline{\underline{a}} \vee \underline{\underline{b}})$ that is found to be the most convenient one, both for theoretical study and for practical applications. Therefore, we shall show how easy it is to prove the equivalence of $(\underline{\underline{a}} \Longrightarrow \underline{\underline{b}})$ and $(\neg \underline{\underline{b}} \Longrightarrow \neg \underline{\underline{a}})$ starting from (8.25) in which the former is equated to $(\neg \underline{\underline{a}} \vee \underline{\underline{b}})$. The steps in the derivation is as follows :

$$(\underline{\underline{a}} \Longrightarrow \underline{\underline{b}}) = (\underline{\underline{c}} \vee \underline{\underline{d}})\ ;\ \underline{\underline{c}} = \neg \underline{\underline{a}}\ ,\ \underline{\underline{d}} = \underline{\underline{b}} \qquad \text{(a)}$$

$$(\neg \underline{\underline{b}} \Longrightarrow \neg \underline{\underline{a}}) = (\underline{\underline{c}}' \vee \underline{\underline{d}}')\ ,\ c' = \underline{\underline{b}}\ ,\ d' = \neg \underline{\underline{a}} \qquad \text{(b)}$$

(applying equation (8.25) again)

(8

But

$$(\underline{\underline{c}}' \vee \underline{\underline{d}}') = (\underline{\underline{d}}' \vee \underline{\underline{c}}') = (\underline{\underline{c}} \vee \underline{\underline{d}}) \qquad \text{(c)}$$

Hence

$$(\underline{\underline{a}} \Longrightarrow \underline{\underline{b}}) \equiv (\neg \underline{\underline{b}} \Longrightarrow \neg \underline{\underline{a}}) \qquad \text{(d)}$$

(using the equations (a) and (b))

Before we leave this question of the different equivalent forms of $\underline{\underline{a}} \Longrightarrow \underline{\underline{b}}$, we shall consider the last one (iv) in (8.12), namely $\underline{\underline{a}} \not\Longrightarrow \neg \underline{\underline{b}}$. This again is very simple to comprehend, provided we understand clearly what we mean by the symbol $\not\Longrightarrow$ (does not imply). There is a difference between the "not" as applied to the "and" in (8.26) and the "not" as applied to implication in (8.12 (iv)). We may designate the latter as a "weak" negation. The "weak" non-implication ($\not\Longrightarrow$; does not imply) means the following :

$$\left.\begin{array}{l} \text{If } \underline{\underline{a}} \text{ "implies" } \underline{\underline{b}} \ (\underline{\underline{a}} \Longrightarrow \underline{\underline{b}}) , \\ \text{then } \underline{\underline{a}} \text{ "does not imply" } \neg \underline{\underline{b}} \ (\underline{\underline{a}} \not\Longrightarrow \neg \underline{\underline{b}}) \end{array}\right\} \quad (8.28)$$

In effect, the negation is transferred to $\underline{\underline{b}}$ and therefore it is merely a question of saying that, if $\underline{\underline{b}}$ is produced as a result of the implication, then " $\neg \underline{\underline{b}}$ is produced as a result of non-implication " . However, it is not exactly equivalent to saying that " it is not possible that $\underline{\underline{a}}$ is true and $\underline{\underline{b}}$ is not true ". $(\neg(\underline{\underline{a}} \,.\, \neg\underline{\underline{b}}))$. This latter is a "strong" negation, and has the form (8.12(ii)) of the Kathavattu statement. The "weak" negation form really says nothing more than the affirmative form $\underline{\underline{a}} \Rightarrow \underline{\underline{b}}$, as we will show in the next lecture, and therefore we shall try as far as possible not to use the operator $\not\Rightarrow$ for any purpose, but always use the form $\neg(\underline{\underline{a}} \,.\, \neg\underline{\underline{b}})$, which employs the connective "and" , whenever the negative consequences of an implication is to be emphasized.

8.8. Application of implication to make deductions

We have, in a way, been rather formal in the last section 8.6; but this digression was necessary in order to show that, what would appear to be rather very complicated mathematics in (8.12), showing the equivalence of four different statements, is really very simple, and quite easy to understand, applying only commonsense. At the same time, our analysis, in terms of the truth table, has established the fact that the equivalences in (8.12) are also mathematically exact and accurate.

Leaving aside the mathematical approach, let us come back again to reality, and examine the different ways in which implication can be employed in the formation of arguments. For this purpose, we shall give examples of the actual types of connected statements corresponding to each of the different

- 52 -

possible ways of writing " $\underline{\underline{a}}$ implies $\underline{\underline{b}}$ " . We can then see how, effectively, it is possible to deduce different consequences, depending upon the logical form that is used for this purpose.

(a) $\underline{\underline{a}} \Rightarrow \underline{\underline{b}}$

Taking the first one, namely $\underline{\underline{a}} \Rightarrow \underline{\underline{b}}$, we need not give any new examples, because the classical "hill-fire-smoke", and "man-Greek-mortal" belong to this category. Here, something is first asserted, and this one is known to imply a second fact, and the second fact is therefore deduced as a consequence. Stated in symbols, the argument has three parts as below :

$$\underline{\underline{a}} \Rightarrow \underline{\underline{b}} \quad ; \quad a_T \quad ; \quad \text{hence} \quad b_T \qquad (8.29)$$

Applied to the two examples, the three parts of the deductive arguments are :

Smoke implies fire.
There is smoke.
Hence, there must be fire. (8.30)

or

Man implies mortal.
Greek is man.
Hence Greek is mortal. (8.31)

This direct method is known by the Latin name "modus ponens" and is without doubt the most common method of making deductions.

$$\underline{(b)\ \neg \underline{\underline{b}} \Longrightarrow \neg \underline{\underline{a}}}$$

In this case, it is absence of something that implies the absence of something else. Taking the case of "hill-fire-smoke" this will have the form — "absence of fire implies absence of smoke", which is equivalent to "smoke implies fire". This type of deduction is called "**modus tollens**", and is summarised by

- 54 -

the sequence :

$$\underline{\underline{a}} \Longrightarrow \underline{\underline{b}} \; ; \; b_F \; ; \text{ hence } a_F \qquad (8.32)$$

Applied to the two examples, the deduction takes the form :

Smoke implies fire ;
There is no fire ;
Hence there will be no smoke (8.33)

Man implies mortal.
Stone is not mortal
Hence stone is not man (8.34)

(c) $\neg(\underline{\underline{a}} \cdot \neg\underline{\underline{b}})$

This form asserts that, if $\underline{\underline{a}}$ implies $\underline{\underline{b}}$, then it is not possible to have simultaneously both $\underline{\underline{a}}$ and $\neg\underline{\underline{b}}$. This form is definitely different from an <u>implication</u>; hence an immediate deduction is not possible. But, we will see from the two examples that the consequences of both (a) and (b)

- 55 -

are deducible from this, given the appropriate initial information.

In the two examples we have taken, this form (c) takes the following shape :

Hill-smoke-fire

The presence of smoke and the absence of fire occurring together is impossible (8.35a)

Hence,

if smoke exists, fire exists;
Also,
if fire does not exist, smoke does not exist (8.35b)

Man-Greek-mortal

To be a man and not a mortal is impossible (8.36a)

Hence,

if X is man, then X is mortal;
and (8.36b)

(d) $\neg \underline{\underline{a}} \vee \underline{\underline{b}}$

This form says $\neg \underline{\underline{a}}$ or $\underline{\underline{b}}$ is always true. As already mentioned, a practical statement based on this is more difficult to comprehend than (a) , (b) or (c) , unless we already appreciate the "inclusive or" . We shall still state the two examples in this form :

Hill-smoke-fire

Absence of smoke, or presence of fire, is always true (8.37)

Hence, if the first part is false, and smoke exists, the second must be true, ~~or~~ i.e. fire exists. (8.38)

Man-Greek-mortal

Being not a man or being mortal is ever true (8.39)

Hence, if Greek is man, (and the first part is false, the second part is true, and) he is mortal. (8.40)

We see that, although the full impact, or power, of the statements (8.37) and (8.39), are not immediately obvious, they still lead to the <u>modus</u> <u>ponens</u> form when properly interpreted. This form (d) $\neg \underline{\underline{a}} \vee \underline{\underline{b}}$ is very useful for application in logical theory, although it is not so convenient for use in real arguments.

However, a modification of (d) into a form using the "or" of two <u>assertions</u> can indeed be useful, when we find that the negation of one implies the assertion of the other, and <u>vice</u> <u>versa</u>.

<u>(e) $\underline{\underline{a}} \vee \underline{\underline{b}}$ and its equivalent implication</u>

Take the statement: "Mr X must have been in Delhi, or Hyderabad, yesterday, since he arrived in Bangalore by the Delhi-Hyderabad-Bangalore plane". Let " X was in Delhi"

- 58 -

be $\underline{\underline{d}}$ and " X was in Hyderabad" be $\underline{\underline{h}}$. We then have "$\underline{\underline{d}} \vee \underline{\underline{h}}$ is true" as the given statement. Applying this, it follows that if he was not in Delhi, then he was in Hyderabad, and <u>vice versa</u>. In symbols, we thus have the equivalences :

$$(\underline{\underline{d}} \vee \underline{\underline{h}}) \equiv (\neg \underline{\underline{d}} \Rightarrow \underline{\underline{h}}) \equiv (\neg \underline{\underline{h}} \Rightarrow \underline{\underline{d}}) \qquad (8.41)$$

It is interesting to note that negative information in the form of X's <u>absence</u> in Delhi, or in Hyderabad, <u>leads to</u> a positive <u>inference</u> about his <u>presence</u> in the other city. Incidentally, there is no objection to his having been both in Delhi and in Hyderabad yesterday; for he could have taken the morning plane from Delhi to Hyderabad and come to Bangalore by the evening plane from Hyderabad and the statement " $\underline{\underline{d}} \vee \underline{\underline{h}}$ " does not preclude this. But, the two implications in (8.41) are still valid. In words, therefore, the equivalence in (8.41) can be stated as follows :

- 59 -

If two facts are together always known to be true, then, if either is given to be false, the other one can be deduced to be true. (8.42)

If now, we make one of the facts a negation in (8.42), we can immediately deduce the equivalences

$$(\underline{\underline{a}} \Rightarrow \underline{\underline{b}}) \equiv (\neg \underline{\underline{a}} \vee \underline{\underline{b}}) \equiv (\neg \underline{\underline{b}} \Rightarrow \neg \underline{\underline{a}}) \quad \text{of (8.27).}$$

Perhaps this route via (8.41) may be the best way of remembering the relation between disjunction and implication. (See also the next chapter, and the design of the tertiary I in the logic machine described in Chapter).

(f) $\neg(\underline{\underline{a}} \, . \, \underline{\underline{b}})$ and its equivalent implication

Unlike (e), this has the form of a negation of a conjunction and is an analogue of (c). Here again, we shall start with an example. Suppose we know that "Mr. Y was not both in Delhi and Calcutta yesterday evening, because he is found

- 60 -

in Bangalore today". Using $\underline{\underline{c}}$ for "Y was in Calcutta" and $\underline{\underline{m}}$ for "Y was in Delhi", we ha~~s~~ve, as the starting point, the knowledge $\neg(\underline{\underline{c}} \cdot \underline{\underline{m}})$. This means that, at most, he could have been only in one of the two cities, Madras and Calcutta. Therefore, if we have independent information that he was actually in Calcutta, then he could not have been in Madras last evening, and vice versa. Hence, we have the result (8.43), similar to (8.41) :

$$\neg(\underline{\underline{c}} \cdot \underline{\underline{m}}) \equiv (\underline{\underline{c}} \Rightarrow \neg\underline{\underline{m}}) \equiv (\underline{\underline{m}} \Rightarrow \neg\underline{\underline{c}}) \qquad (8.43)$$

In this case, positive information about his presence in one city leads to an inference about his absence in the other. In words, (8.43) can be stated as follows :

> If two facts cannot both be true, then, if either is given to be true, the other one can be deduced to be false. (8.44)

(This is exactly the opposite of the example (e), where falsity of one implies the truth of the other.)

Once again, if we make one of the two facts to be a negation in (8.44), we obtain at once the equivalences,

$$(\underline{\underline{a}} \Rightarrow \underline{\underline{b}}) \equiv \neg(\underline{\underline{a}} \cdot \neg\underline{\underline{b}}) \equiv (\neg\underline{\underline{b}} \Rightarrow \neg\underline{\underline{a}}) ,$$

given in the Kathavattu argument and illustrated in (c). Here again, the use of (8.43) seems to be the best way of appreciating the relation between conjunction (in the form of nand) and implication. The reason why only "nand" occurs (and not "and") will become clear in the next chapter, when we discuss the matrix formalism.

Perhaps one final comment may be relevant, regarding this Lecture 8. It uses only commonsense, and some small applications of the ternary truth table 8.2(b), and, in this sense, everything in this chapter is written within the framework of Classical Logic. Our SNS system is involved only in the occurrence of the D state in a few equations summarizing the classical logic results, and in Table 8.2(a) for the binary form of the implication connective. These have not been applied anywhere in working out results and formulae in this lecture, so that every consequence of the implication statement $\underline{\underline{a}} \Rightarrow \underline{\underline{b}}$, which we have stated **here, is** understandable in terms of standard classical logic.

In the next lecture, we shall not only introduce the form of the matrix representing implication, but we shall also work out every type of pattern of "deductive sequences" that is possible in an argument. This will be done by examining the consequences of all possible ways of attaching a "not" to the various input and output statements. The great beauty of this approach is that we need remember only one rule —— namely what is meant by "modus ponens" discussed in the subsection (a) above.

All the rest can be reduced to this one <u>via</u> routine matrix algebra that requires no intelligence, but only bullwork (something that a computer can do). Logic is thus reduced to commonsense + computer.

F O R T R A N   P R O G R A M   F O R   S E N T E N T I A L   L O G I C*

G.N. Ramachandran and T.A. Thanaraj

Mathematical Philosophy Group
Indian Institute of Science
Bangalore 560 012.

*MATPHIL Reports No. 12, September 1980

CONTENTS

.. Contd.

FORTRAN PROGRAMS FOR SENTENTIAL LOGIC

G.N. Ramachandran and T.A. Thanaraj

Abstract:- Although the relation between logic and Boolean algebra is well established, computer programs for working logical problems are not available. We have developed an extension of Boolean algebra in terms of two-element Boolean vectors and 2 x 2 Boolean matrices which is very versatile and provides algorithms for almost all aspects of sentential logic. The matrix algebra is also expressible in terms of classical connectives like 'Not', 'And', 'Or', 'Nor', 'Xor', etc. In terms of these algorithms, procedures are developed for solving logical problems, which utilize "logical graphs", joining the various terms via logical connectives, for writing out a set of equations, which are then solved. It is believed that the theory is practically complete, and a few examples are also given of FORTRAN problems in logic.

Index Terms:- Algebra for logic, Algorithms for arguments, States and connectives, Reverse operators in logic, Logical graphs.

..2

FORTRAN FOR LOGIC

I. Introduction

The subject of sentential logic, or propositional logic as it is called otherwise, is well discussed in most books on logic (see e.g. [1], [2], [3]). The principles governing the truth values of terms, or statements, and modifications which they undergo when they are operated upon by logical connectives are also discussed therein. The logic is based on a two-valued truth function, which can take the logical states "true" or "false", which can also be represented by the Boolean elements 1 and 0. The connection between the algebra of logical connectives in this "Classical Logic" (CL), as we shall call it, and the Boolean algebra composed of two elements 1 and 0, is well known [4]. We shall take this for granted in what follows. Every equation in logic, such as $\underline{\underline{a}}$ and $\underline{\underline{b}}$ = $\underline{\underline{c}}$, $\underline{\underline{a}}$ or $\underline{\underline{b}}$ = $\underline{\underline{c}}$ etc., has a counterpart in the Boolean algebra where "and" and "or" are replaced by the operations of Boolean multiplication and Boolean addition, which also correspond to the concepts of intersection ($\cap$) and union ($\cup$) of sets in classical set theory.

When we attempted to write FORTRAN programs for general problems in propositional logic, we discovered that the number of operators that are defined in classical logic, and even the number of truth values as defined therein, are insufficient for

an efficient manipulation of these. It is not our contention that there is any defect in the well-known logical theory of two-valued logic as it occurs in propositional logic; but only that it can be enlarged and improved upon. We are only attempting to modify and symbolise it in such a manner that the manipulations involving logical operations, for solving problems in general, are made as simple and as universal as possible. In trying to do this, we arrived at two or three interesting results, which we give right here for an illustration of the scope and range of our approach. The reasons for making these modifications and their capabilities are briefly discussed in the sections that follow.

(i) The two truth values "true" and "false" have to be replaced by four truth values: true ($\underline{\underline{T}}$), false ($\underline{\underline{F}}$), doubtful ($\underline{\underline{D}}$) and impossible ($\underline{\underline{X}}$). In effect, this is done by replacing a single-component Boolean element to represent the truth value of terms by a double component Boolean vector ($\alpha\quad\beta$), which obviously has four possible states. This is discussed in Section II.

(ii) In addition to the operators such as E(equals), N(negates) A(and), O(or), I(implies) etc., which may be called "forward" or "direct" operators, we find the need to define suitably the "reverse" operators corresponding to these, which may be symbolized by $\overleftarrow{E}$, $\overleftarrow{N}$, $\overleftarrow{A}$, $\overleftarrow{O}$, $\overleftarrow{I}$ etc. These operators have truth tables which are quite normal, but which are not discussed in any book or

..4

references to our knowledge. What is more interesting is that some of these operators, particularly the reverse operators $\overleftarrow{A}$ and $\overleftarrow{O}$, for "reverse and" and "reverse or", automatically require the postulation of the four kind of truth states mentioned above.

(iii) Two new operators W(with) and U(upon) are also found to be needed, of which W is indirectly employed in almost every logical argument in classical logic. It deals with the comparison of the truth values for the same statement as obtained from two different pathways. It can give a positive result if the two are consistent, but can also point to an inconsistency, if one path leading <u>via</u> W gives $\underline{\underline{T}}$, and the other path $\underline{\underline{F}}$, which are obviously contradictory to each other. The nature of this operator W and its use is discussed in Section IV. We have named the logic (with the above said modifications in CL) as SYĀD-NYĀYA-SYSTEM (SNS). (Syād means "may-be", Nyāya means "logic", in Sanskrit).

In the subsequent sections, the algebra of this approach is very briefly given and then the technique of converting it into a FORTRAN program is discussed schematically, with samples of the sub-routines given for selected examples. A general survey of the whole of this approach is given in two reports by the senior author ([5], [6]), which contain fuller accounts of the matrix

..5

algebra. We shall often refer to these papers for details, although all the essential ideas are briefly given in this paper so as to make it self-contained. The earlier papers, however, do not contain anything about the development of the FORTRAN program.

1. Logical Graphs and Introduction to their Use

Using the above concepts, it is possible to convert any logical argument consisting of a number of individual 'sentences' to a suitably defined graph which have terms and operators as nodes. A node corresponding to a term, in general, has one or two paths meeting at it, while an operator may have two, three, or more paths entering into, and one path leaving, the corresponding node. The example of a very simple problem is shown graphically in Fig.1 and the corresponding logical equations are given below in terms of classical logic, including the new operators W and U. The nature of this logical representation and the need for W and U become immediately obvious on looking at this example. The problem reads as follows:

"A detective has been detailed by a client to find out if two persons P and Q have met and concluded a financial deal at 1 p.m. on a particular day. A third person R is trying his best to follow the pair and be with one, or the other, such that the two never have an occasion to be together alone for completing

..6

the negotiations, without his being also on the spot with them.

The scene is in Hotel X, and the master detective has employed three reliable agents A, B, C. Agent A reports to him that he saw either P or Q continuously in the hotel during the important hours between 12 p.m. and 2 p.m. and that both were never away from the site at the same time. Agent B reports that, to the best of his knowledge, P was in the hotel at the appropriate time and R was not with him. Agent C says that only one of R or Q was in the hotel at that particular time."

"Problem: Can you deduce from these, as the detective did that the deal was concluded?"

Using the symbols, N for "not", A for "and", O for (inclusive) "or" and XO for "exclusive or", we have the following logical equations for $\underline{a}$, $\underline{b}$, $\underline{c}$ (standing for the statements made by A, B, C), where $\underline{p}$, $\underline{q}$, $\underline{r}$ stand for the presence of P, Q, R respectively at the appropriate time for the meeting.

$$\underline{p} \ \mathrm{O} \ \underline{q} = \underline{a} \qquad (1a)$$

$$\underline{p} \ \mathrm{A} \ (\underline{r} \ \mathrm{N}) = \underline{b} \qquad (1b)$$

$$\underline{q} \ \mathrm{XO} \ (\underline{r}) = \underline{c} \qquad (1c)$$

..7

2. Normal procedure for a computer

If one normally puts this problem to a computer, one would put in the truth tables for N, A, O and XO, and use as inputs all possible combinations (namely 8) of $\underline{\underline{T}}$ and $\underline{\underline{F}}$ for $\underline{\underline{p}}$, $\underline{\underline{q}}$ and $\underline{\underline{r}}$, to find out which combination gives $\underline{\underline{a}} = \underline{\underline{T}}$, $\underline{\underline{b}} = \underline{\underline{T}}$, and $\underline{\underline{c}} = \underline{\underline{T}}$. This may be graphically illustrated by the left half of Fig.1, showing the path of transfer of information by full lines.

Such a truth table is shown in Table 1, and it will be seen that the only set of truth values for $\underline{\underline{p}}$, $\underline{\underline{q}}$, $\underline{\underline{r}}$ which leads to $\underline{\underline{T}}$ for all three of $\underline{\underline{a}}$, $\underline{\underline{b}}$, $\underline{\underline{c}}$, is $\underline{\underline{T}}$, $\underline{\underline{T}}$, $\underline{\underline{F}}$, i.e. $\underline{\underline{p}}$ and $\underline{\underline{q}}$ were present and $\underline{\underline{r}}$ was absent, which is precisely the condition necessary for the deal to be concluded.

In the electronic analog computer which we have constructed, and which we have discussed in an accompanying paper (8), there is provision to check for the outputs $\underline{\underline{a}}$, $\underline{\underline{b}}$, $\underline{\underline{c}}$ to be $\underline{\underline{T}}$, by using the module G for the logical operator "agree" (see IV-7 below). In the second half on Fig.1, three such are included, which will give the output $\underline{\underline{T}}$ if the input is $\underline{\underline{T}}$, and $\underline{\underline{F}}$ if it is any of $\underline{\underline{F}}$, $\underline{\underline{D}}$ or $\underline{\underline{X}}$. We can also test that <u>all the three G's</u> unanimously give yes signals by using the module for the "unanimity" operator, U

..8

TABLE 1: TRUTH TABLE FOR PROBLEM 1, USING THE LOGICAL GRAPH SHOWN IN FIGURE 1

P	Q	R	A	B	C	X
T	T	T	T	F	F	D
T	T	F	T	T	T	T
T	F	T	T	F	T	D
T	F	F	T	T	F	D
F	T	T	T	F	F	D
F	T	F	T	F	T	D
F	F	T	F	F	T	D
F	F	F	F	F	F	F

..9

(see IV-2), and inputting the three signals coming from the C's into the module U. Then, the output x of U being T is an indication that all the requisite conditions of the problem are simultaneously satisfied. (Note that x is T only for p = T, q = T, r = F in row 2 of Table 1).

Fig.1. Logical graph representing Problem 1 and its possible routine solution.

One simply wires the interconnections between p, q and r and the different modules as shown in Fig.1, and looks at the output x of U, for the 8 possible combinations of input. Then, it is found that it is green (showing that a, b, c are all T) only for p = T, q = T, r = F. Hence, we deduce, in reverse, that the observations of the detectives A, B, C mean uniquely that P and Q were present, and R absent at the time of the meeting.

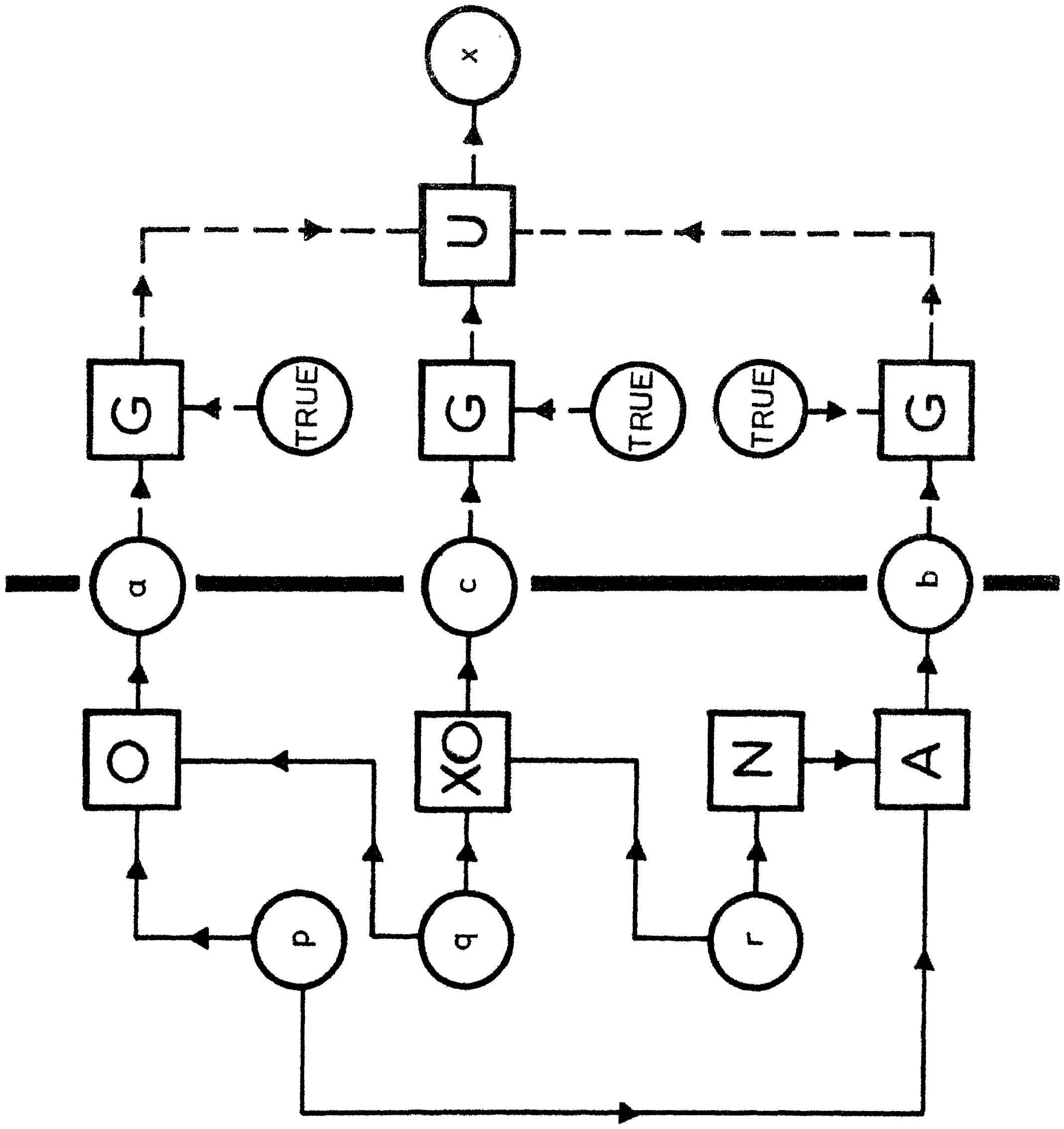
3. Shortened procedure simulating the human approach

A very brief proof of the desired result can however be given as follows:

(a) Since "p and ¬r" is true from (1b), p = T and r = F

(b) Since q and r have opposite truth values from (1c) and r = F from above, it follows that q = T. Hence,

..10


x
U
G
TRUE
G
TRUE
TRUE
G
a
c
b
O
XO
N
A
p
q
r

p = T, q = T, r = F.

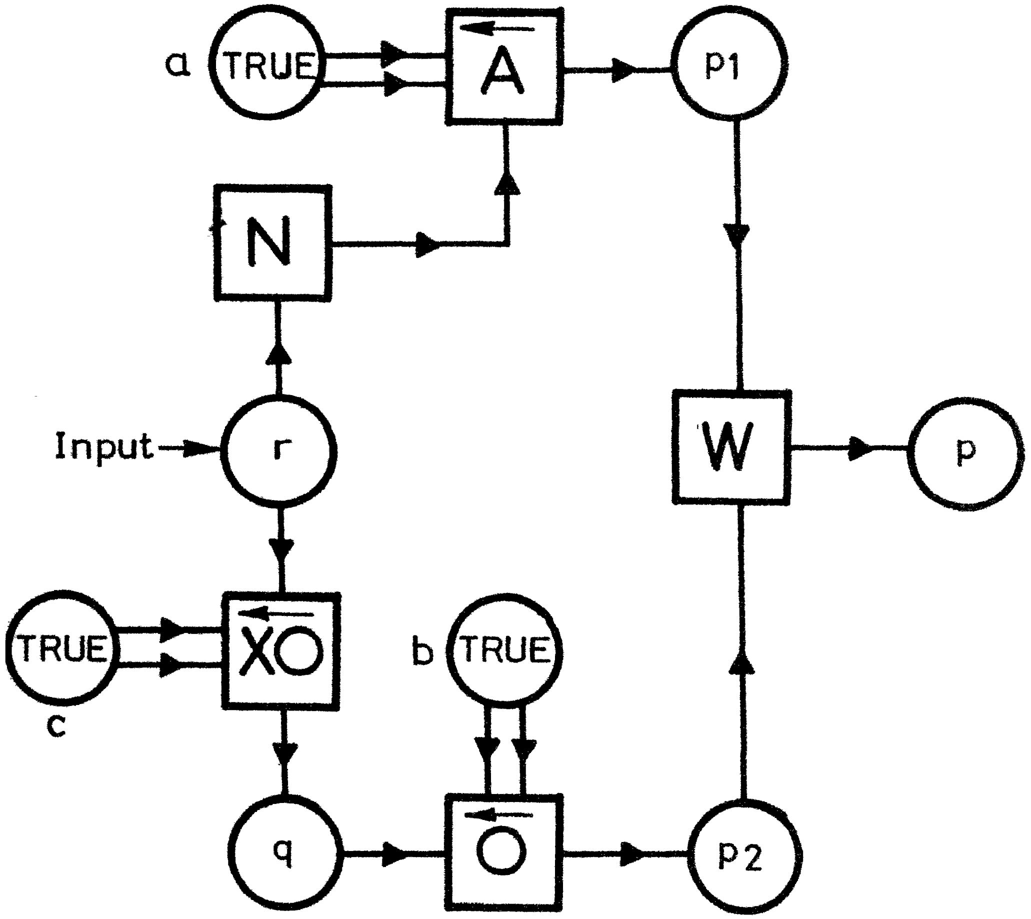
(c) The combination of T, T for p , q is consistent with Eqn.(1a) also.

Obviously, the computer must be made to simulate this thinking process to be efficient. We found that the best way of doing this, is to use the "reverse" operators $\overleftarrow{A}$, $\overleftarrow{O}$ and $\overleftarrow{XO}$. Of these, $\overleftarrow{XO}$ = XO itself, and we obtain the simple graph shown in Fig.2 with just one input (r), the three reverse operators mentioned above, and a W for the consistency check. The logical steps are also given in Fig.2.

Fig.2. Logical graph of Problem 1 reformulated, requiring only one input r

The truth table for this way of solving the problem represented by the graph in Fig.2, is shown in Table 2, using only one independent input r. There are only two trials to be made, for r = T and F respectively. One gives rise to an inconsistency for p, while the other gives the correct answer (T, T, F) for (p, q, r). The computer itself has been instructed to check for freedom from inconsistencies, and print such a solution at the bottom. (It may be mentioned that the results for a typical example has been included in this introductory section itself, following the advice received from Prof.

..12



Reformulation of Problem 1

$\underline{\underline{T}}\ \overleftarrow{A}\ (\underline{\underline{r}})N \ = \ \underline{\underline{p1}}$

$\underline{\underline{T}}\ \overleftarrow{XO}\ \underline{r} \ = \ \underline{q}$ *

[illegible] $\overleftarrow{}$ $\underline{\underline{q}} \ = \ \underline{\underline{p2}}$

$\underline{\underline{p1}}$ [illegible] $\underline{\underline{p2}} \ = \ \underline{p}$

* This is equivalent to
$(\underline{r})N \ = \ \underline{\underline{q}}$

Fig. 2. Logical graph of Problem 1 reformulated to simulate the " human" method of solution.

TABLE 2: SHORTER TRUTH TABLE USING ONLY ONE INPUT*

R	(R)N	P1	Q	P2	P
T	F	X	F	T	X
F	T	T	T	D	T

*THE SOLUTION WITHOUT A CONTRADICTION IS P = T , Q = T , R = F

..14

Sir Lawrence Bragg by the senior author in 1948, when he wrote a paper containing lengthy mathematics, but dealing with a real practical problem. He was told to "Cut the cackle and come to the horses".)

II. Truth values, Logical variables and Operators.

1. Truth values

As mentioned earlier, the states, or truth values, possible in our new system (SNS) of logic are true ($\underline{\underline{T}}$), False ($\underline{\underline{F}}$), Doubtful ($\underline{\underline{D}}$), and Contradiction ($\underline{\underline{X}}$). These truth values can be defined in terms of a two-element Boolean vector ($\alpha \quad \beta$), where α, β = 1 or 0. The component α represent the probability for truth and β the probability for falsehood. Hence the state vector for $\underline{\underline{T}}$ is (1 0) and for $\underline{\underline{F}}$, (0 1). The state $\underline{\underline{D}}$ is one in which the logical statement could be either true or false, which may arise either from lack of data, or as a consequence of the previous step in the argument, which leads to both the states $\underline{\underline{T}}$ and $\underline{\underline{F}}$ being possible. The Boolean addition of the state vectors of $\underline{\underline{T}}$ and $\underline{\underline{F}}$ leading to

$$(1 \quad 0) \oplus (0 \quad 1) = (1 \oplus 0 \quad 0 \oplus 1) = (1 \quad 1) \quad (2$$

satisfies these conditions, and (1 1) is therefore appropriate

..15

for $\underline{\underline{D}}$ (A more rigorous argument will be found in the book [7] being written by the senior author). Similarly, the state of impossibility $\underline{\underline{X}}$ often arises from the superposition of two arguments which yield simultaneously the truth values true and false for it. Hence, its state vector, $\underline{\underline{X}}$, can be obtained by the Boolean multiplication of the state vectors of $\underline{\underline{T}}$ and $\underline{\underline{F}}$, so that

$$\underline{\underline{X}} = (1 \quad 0) \otimes (0 \quad 1) = (1 \otimes 0 \quad 0 \otimes 1) = (\qquad (2$$

The doubtful state is never used as an initial input into an argument because the corresponding statement does not state any thing definite. However, $\underline{\underline{D}}$ can occur as the output of the previous step during the progress of an argument, but it may be converted into either $\underline{\underline{T}}$ or $\underline{\underline{F}}$ by suitable combinations with other terms <u>via</u> connective operators, like "and" and "or". The state $\underline{\underline{X}}$, on the other hand, indicates a contradiction in the arguments leading to it when it occurs, and therefore one cannot proceed further from thereon. However, we can go back and examine the validity of the earlier steps, using this information. Further, if $\underline{\underline{X}}$ is used as an input, it would always lead to an $\underline{\underline{X}}$ in the next step generated by it <u>via</u> any connective and it cannot be removed. (There is one exception, for SNS operators only, which are discussed in [7], but it is not important for the discussion in this paper).

..16

2. Boolean variables: Any general term has a truth value of one of these four constants $\underline{\underline{T}}$, $\underline{\underline{F}}$, $\underline{\underline{D}}$, $\underline{\underline{X}}$ and therefore, a term $\underline{\underline{a}}$ is represented by the two component vector $(\underline{a}_\alpha \quad \underline{a}_\beta)$ where $\underline{a}_\alpha$ and $\underline{a}_\beta$ can both be either 0 or 1. The matrix algebra of such Boolean vectors with two components has been explored much more in detail in an earlier report from this laboratory [5] and also in [7]. Therefore, no details are given of the technique, but only the results.

3. Operators: These are of two types - unary and binary. The simplest example of a unary operator is in the equation

$$\underline{\underline{a}}\,E = \underline{\underline{b}} \quad (\text{equivalence, } \underline{\underline{a}} = \underline{\underline{b}}) \tag{3}$$

and

$$\underline{\underline{a}}\,N = \underline{\underline{b}} \quad (\text{negation, } \neg\underline{\underline{a}} = \underline{\underline{b}}) \tag{4}$$

The corresponding equations connecting the two elements $(\underline{a}_\alpha \quad \underline{a}_\beta)$ of $\underline{\underline{a}}$ and the two elements of $\underline{\underline{b}}$ are

$$(\underline{\underline{a}}\,E = \underline{\underline{b}}) \equiv (\underline{a}_\alpha = \underline{b}_\alpha\,,\ \underline{a}_\beta = \underline{b}_\beta) \tag{5a}$$

$$(\underline{\underline{a}}\,N = \underline{\underline{b}}) \equiv (\underline{a}_\beta = \underline{b}_\alpha\,,\ \underline{a}_\alpha = \underline{b}_\beta) \tag{5b}$$

The α and β components obey the rules of classical logic (CL). They take only the classical truth values TRUE (1) and FALSE (0) and they are acted upon by the classical logical connectives

..17

namely EQU, NOT, OR, XOR, AND. Everything in SNS logic can be given in terms of the classical logical connectives applied to the components of our two element vector.

Two typical examples, in the case of binary operators, are "or" (O) and "and" (A), for which has the SNS equations may be written in the form

$$\underline{\underline{a}}\ O\ \underline{\underline{b}} = \underline{\underline{c}} \tag{6a}$$

$$\underline{\underline{a}}\ A\ \underline{\underline{b}} = \underline{\underline{c}} \tag{6b}$$

It is readily verified that these are equivalent to the classical equations

$$O : \underline{a}_{\alpha} \text{ OR } \underline{b}_{\alpha} = \underline{c}_{\alpha} \quad ; \quad \underline{a}_{\beta} \text{ AND } \underline{b}_{\beta} = \underline{c}_{\beta} \tag{7a}$$

$$A : \underline{a}_{\alpha} \text{ AND } \underline{b}_{\alpha} = \underline{c}_{\alpha} \quad ; \quad \underline{a}_{\beta} \text{ OR } \underline{b}_{\beta} = \underline{c}_{\beta} \tag{7b}$$

III. FORTRAN Implementation

In FORTRAN, the logical values available are .TRUE. and .FALSE. and the logical operators are .NOT., .AND. , .OR., .XOR. and .EQU. (the last two operators are allowed in DEC-10 implementation). First, two function subprograms A(a term) and B(a term), e.g. A(VA), B(VA), are defined to give the classical truth values of the α and β components of the relevant term,

..18

which will be one of the above-said classical logical values. Table 3 gives the values of the functions A and B for the four SNS states used as logical constants.

Table 3

State of VA	A(VA)	B(VA)
$\underline{\underline{T}}$	.TRUE.	.FALSE.
$\underline{\underline{F}}$	.FALSE.	.TRUE.
$\underline{\underline{D}}$	.TRUE.	.TRUE.
$\underline{\underline{X}}$	.FALSE.	.FALSE.

Another function subprogram of general utility is STATE(α component, β component), used to obtain the SNS truth value, when we are given α and β components. Using these three defined functions A, B and STATE, and the classical logical operators NOT, AND, etc., all the subprograms for the SNS connectives have been written. The following FORTRAN symbology has been developed for this purpose.

..19

1. Constants and Variables

The SNS logical constants, $\underline{\underline{T}}$, $\underline{\underline{F}}$, $\underline{\underline{D}}$, $\underline{\underline{X}}$ are called STR, SFL, SDF, SXX, declared in the form

DATA STR, SFL, SDF, SXX / 'T', 'F', 'D', 'X' /

The SNS logical variables will start with the letter V. Examples are VA, VB, VMAN, VX(1), VX(2), etc.

The α and β components will start with A and B respectivel Examples are AVA, BVA; AVB, BVB; AVMAN, BVMAN; etc. For this purpose, the characters A and B are implicitely declared as logical variables.

SNS logical variables obtain their logical values either by assignment - for example VA = 'T' , or VA = STR, or VB = SFL , etc., or by reading a data line that just contains T, F, D, or X as the case may be, and not in quotes.

2. Connectives

The unary logical connectives start with Z followed by the one-letter logical symbol, e.g. ZE for equivalence, ZN for negati

The binary connectives also start with Z, but have three letters. the second is F, for forward, or R, for reverse,

..20

operators (see V), and the third letter is the one-letter symbol of the operator. e.g. ZFA for forward "and", ZRO for reverse "or"

3. __Negation function__ - N

Using the above symbology, the logical equation $\underline{\underline{a}}$ N = $\underline{\underline{b}}$ becomes the FORTRAN statement VB = ZN(VA), and the function ZN(VA) is defined as

```
AVB  =  B(VA)
BVB  =  A(VA)
 ZN  =  STATE(AVB,BVB)
```

which can be simplified to a single statement

```
ZN  =  STATE(B(VA),A(VA))
```

4. __Functions__ - A __and__ O

Similarly, for the operator A, corresponding to the equation $\underline{\underline{a}}$ A $\underline{\underline{b}}$ = $\underline{\underline{c}}$, the FORTRAN statement is

```
VC  =  ZFA(VA,VB)
```

and the function ZFA is defined as

..21

```
AVC  =  A(VA).AND.A(VB)
BVC  =  B(VA).OR.B(VB)
ZFA  =  STATE(AVC,BVC)
```

using the Eq. 7(b). Similarly, for the operator O, correspondin to the equation $\underline{\underline{a}}$ O $\underline{\underline{b}}$ = $\underline{\underline{c}}$, the FORTRAN expression is VC = ZFO(VA,VB), and the function ZFO is defined as follows, using Eq. 7(a)

```
AVC  =  A(VA).OR.A(VB)
BVC  =  B(VA).AND.B(VB)
ZFO  =  STATE(AVC,BVC)
```

IV. Other Direct Operators

1. The operator for consistency - "with"

Suppose there occurs a situation in the middle of the progress of an argument, in which one line of argument starting with a term $\underline{\underline{a}}$ leads to a result $\underline{\underline{c}}$ and there is also another line of argument starting with a term $\underline{\underline{b}}$ that also leads to $\underline{\underline{c}}$. What is the resultant state of $\underline{\underline{c}}$? The answer is obtained by comparing the result of the first argument with that of the second argument for "consistency" and thus coming to a judgement.

..22

This operator which compares one result with another is what we call "with", represented by the symbol W , written in the form, a with $\underline{\underline{b}}$ = $\underline{\underline{c}}$, or

$$\underline{\underline{a}}\ W\ \underline{\underline{b}} = \underline{\underline{c}} \tag{8a}$$

In the above equation, if both $\underline{\underline{a}}$ and $\underline{\underline{b}}$ are $\underline{\underline{T}}$, then $\underline{\underline{c}}$ becomes $\underline{\underline{T}}$. Similarly if both the inputs $\underline{\underline{a}}$ and $\underline{\underline{b}}$ are $\underline{\underline{F}}$, then $\underline{\underline{c}}$ gets the truth value $\underline{\underline{F}}$. If one is $\underline{\underline{T}}$ while the other is only $\underline{\underline{D}}$, then the definite result $\underline{\underline{T}}$ dominates over $\underline{\underline{D}}$ and it prevails to make $\underline{\underline{c}} = \underline{\underline{T}}$. Similarly, if one is $\underline{\underline{F}}$, and the other $\underline{\underline{D}}$, the resultant $\underline{\underline{a}}\ W\ \underline{\underline{b}}$ is $\underline{\underline{F}}$. On the other hand, if the two lines of argument leading to $\underline{\underline{c}}$ give opposite truth values for it, namely $\underline{\underline{T}}$ and $\underline{\underline{F}}$, then there is clearly a contradiction, because it is possible for $\underline{\underline{c}}$ to be $\underline{\underline{T}}$ and $\underline{\underline{F}}$ at the same time. We then assign the truth value $\underline{\underline{X}}$ for $\underline{\underline{c}}$, and call it an "impossible" state, as mentioned earlier. Thus, the truth table for W is as given in Table 4. It is readily shown that the corresponding classical equations from which the properties of W can be built up are extremely simple - namely

$$\underline{a}_{\alpha} \text{ AND } \underline{b}_{\alpha} = \underline{c}_{\alpha} \quad ; \quad \underline{a}_{\beta} \text{ AND } \underline{b}_{\beta} = \underline{c}_{\beta} \tag{8b}$$

..23

Table 4. Truth table for "with" W and "upon" U

a	b	c = a W b	d = a U b
T	T	T	T
F	F	F	F
T	F	X	D
F	T		
D	T	T	D
T	D		
D	F	F	D
F	D		
D	D	D	D

Following the symbology given in section III, the FORTRAN statement for W is

```
VC = ZFW(VA,VB)
```

and the function ZFW can be written as follows, using (8b):

```
AVC  =  A(VA).AND.A(VB)
BVC  =  B(VA).AND.B(VB)
ZFW  =  STATE(AVC,BVC)
```

<u>2. The operator "upon"</u> - U

Analogous to the operator W, we can also define a fourth SNS binary connective "upon" (U) as follows. The SNS equation

$$\underline{\underline{a}}\ U\ \underline{\underline{b}} = \underline{\underline{c}} \qquad (9a)$$

corresponds to the classical equations

$$\underline{a}_{\alpha}\ \text{OR}\ \underline{b}_{\alpha} = \underline{c}_{\alpha}\ ; \quad \underline{a}_{\beta}\ \text{OR}\ \underline{b}_{\beta} = \underline{c}_{\beta} \qquad (9b)$$

Using (9b), we arrive at the truth table for U as in Table 4, which has the following consequences. If the inputs $\underline{\underline{a}}$ and $\underline{\underline{b}}$ are <u>both</u> $\underline{\underline{T}}$ or both $\underline{\underline{F}}$, then $\underline{\underline{c}}$ has also the same state as the two inputs. In all other cases it is $\underline{\underline{D}}$. Thus U may be considered as checking for "unanimity", just as W checks for "consistency", The FORTRAN statement corresponding to (9a) is

```
VC  =  ZFU(VA,VB)
```

..25

and the function ZFU can be written as follows:

```
AVC  =  A(VA).OR.A(VB)
BVC  =  B(VA).OR.B(VB)
ZFU  =  STATE(AVC,BVC)
```

3. Binary Implies - I

The binary logical connective "implies" (I) in the equation

$$\underline{\underline{a}}\ \mathrm{I}\ \underline{\underline{b}} = \underline{\underline{c}} \qquad (10a)$$

is very well known in logic. The truth table for I for the classical states $\underline{\underline{T}}$ and $\underline{\underline{F}}$ is given in Table 5(a),

Table 5. Truth tables for I and J

(a) I : $\underline{\underline{a}}$ implies $\underline{\underline{b}}$

$\underline{\underline{a}}$ \ $\underline{\underline{b}}$	$\underline{\underline{T}}$	$\underline{\underline{F}}$
$\underline{\underline{T}}$	$\underline{\underline{T}}$	$\underline{\underline{F}}$
$\underline{\underline{F}}$	$\underline{\underline{T}}$	$\underline{\underline{T}}$

(b) J : $\underline{\underline{a}}$ implicates $\underline{\underline{b}}$

$\underline{\underline{a}}$ \ $\underline{\underline{b}}$	$\underline{\underline{T}}$	$\underline{\underline{F}}$
$\underline{\underline{T}}$	$\underline{\underline{T}}$	$\underline{\underline{T}}$
$\underline{\underline{F}}$	$\underline{\underline{F}}$	$\underline{\underline{T}}$

..26

and, as is shown in all books on logic (e.g. [1], [2], [3]), I is expressed in terms of A or O as follows:

$$\underline{\underline{a}}\ I\ \underline{\underline{b}} \iff \neg\underline{\underline{a}}\ O\ \underline{\underline{b}} \iff \neg(\underline{\underline{a}}\ A\ \neg\underline{\underline{b}}) \qquad (10b)$$

The corresponding classical equations for the components $\underline{c}_\alpha$ and $\underline{c}_\beta$ can, therefore, be written, in the simple form,

$$\underline{a}_\beta \text{ OR } \underline{b}_\alpha = \underline{c}_\alpha \quad ; \qquad \underline{a}_\alpha \text{ AND } \underline{b}_\beta = \underline{c}_\beta \qquad (10c)$$

The FORTRAN statement corresponding to (10a) is

```
VC = ZFI(VA,VB)
```

It can be derived using (10b) and the subprogram ZFI is then straightforward:

```
ZFI = ZFO(ZN(VA),VB)
```

4. Binary Implicates - J

It is obvious that $\underline{\underline{a}}\ I\ \underline{\underline{b}} = \underline{\underline{c}}$ is true means that $\underline{\underline{a}}$ is a sufficient condition for $\underline{\underline{b}}$ to be true. In the same way, we can also think of $\underline{\underline{a}}$ being a necessary condition for $\underline{\underline{b}}$ to be true. In this case, we shall say that "$\underline{\underline{a}}$ implicates $\underline{\underline{b}}$", and the symbol J is used for it. The logical equation for J is written as

$$\underline{\underline{a}}\ J\ \underline{\underline{b}} = \underline{\underline{c}} \qquad (11a)$$

..27

The truth table for this operator J is shown in Table 5(b) and, we have, analogous to (10b),

$$\underline{\underline{a}}\ J\ \underline{\underline{b}} \iff \neg\underline{\underline{b}}\ O\ \underline{\underline{a}} \iff \neg(\underline{\underline{b}}\ A \neg\underline{\underline{a}}) \qquad (11b)$$

The corresponding classical equations are also very simple, namely -

$$\underline{\underline{b}}_{\beta}\ \text{OR}\ \underline{\underline{a}}_{\alpha} = \underline{\underline{c}}_{\alpha}\ ; \qquad \underline{\underline{b}}_{\alpha}\ \text{AND}\ \underline{\underline{a}}_{\beta} = \underline{\underline{c}}_{\beta} \qquad (11c)$$

As for I, the FORTRAN statement corresponding to J is

```
VC = ZFJ(VA,VB)
```

and the subprogram for ZFJ is

```
ZFJ = ZFO(VA,ZN(VB))
```

It should be noted that

$$\underline{\underline{a}}\ J\ \underline{\underline{b}} \iff \underline{\underline{b}}\ I\ \underline{\underline{a}} \qquad (11d)$$

5. Unary If - Y

In addition to the binary form I, we have found it useful to have the unary relation - "If $\underline{\underline{a}}$, then $\underline{\underline{b}}$". The "if" connective is represented by the symbol Y (the first letter of yadi, in Sanskrit, = if), and the logical relation is expressed by

$$\underline{\underline{a}}\ Y = \underline{\underline{b}} \qquad (12a)$$

..28

The truth table for Y is as shown in Table. 6(a)

Table 6. Truth Table for Y and V

(a) Y : "If ... then"

$\underline{\underline{a}}$	$\underline{\underline{b}}$
$\underline{\underline{T}}$	$\underline{\underline{T}}$
$\underline{\underline{F}}$	$\underline{\underline{T}}$ or $\underline{\underline{F}}$ ($\equiv \underline{\underline{D}}$)
$\underline{\underline{D}}$	$\underline{\underline{D}}$

(b) V : "Only if"

$\underline{\underline{a}}$	$\underline{\underline{b}}$
$\underline{\underline{T}}$	$\underline{\underline{T}}$ or $\underline{\underline{F}}$ ($\equiv \underline{\underline{D}}$)
$\underline{\underline{F}}$	$\underline{\underline{F}}$
$\underline{\underline{D}}$	$\underline{\underline{D}}$

The most interesting consequence of this truth table is that $\underline{a}_{\underline{\underline{T}}}$ corresponds $\underline{b}_{\underline{\underline{T}}}$, but $\underline{a}_{\underline{\underline{F}}}$ leads to nothing definite for $\underline{\underline{b}}$, but only to a doubtful state $\underline{b}_{\underline{\underline{D}}}$. This follows from the truth table for I (implies) in $\underline{\underline{a}}$ I $\underline{\underline{b}}$ = $\underline{\underline{c}}$, and taking $\underline{\underline{c}}$ to be true. Hence, $\underline{\underline{a}}$ Y = $\underline{\underline{b}}$ means that "if $\underline{\underline{a}}$ is true, then $\underline{\underline{b}}$ is true", and nothing more follows. These are expressed by the classical equations

$$\underline{a}_{\alpha} \text{ OR } \underline{a}_{\beta} = \underline{b}_{\alpha} \quad ; \quad \underline{a}_{\beta} = \underline{b}_{\beta} \qquad (12b)$$

for the logical relation $\underline{\underline{a}}$ Y = $\underline{\underline{b}}$.

..29

The FORTRAN statement of (12a) is

```
VB = ZY(VA)
```

and the subprogram for ZY is

```
AVB = A(VA).OR.B(VA)
BVB = B(VA)
 ZY = STATE(AVB,BVB)
```

6. Unary Only If – V

The unary "only if" connective is represented by the symbol V (using the first consonant in the word avasya, in Sanskrit, = necessary). The logical relation is written as

$$\underline{\underline{a}}\ V = \underline{\underline{b}} \qquad (13a)$$

This means "$\underline{\underline{a}}$ is implied by $\underline{\underline{b}}$", or truth of $\underline{\underline{a}}$ is necessary for truth of $\underline{\underline{b}}$. We can arrive at the truth table for V from J , and this is shown in Table 6(b).

The classical equations corresponding to (13a) are

$$\underline{a}_{\alpha} = \underline{b}_{\alpha} \quad ; \quad \underline{a}_{\alpha} \text{ OR } \underline{a}_{\beta} = \underline{b}_{\beta} \qquad (13b)$$

On comparing the Tables 7(a) and 7(b), and comparing the equations (12b) and (13b), we can see that

$$(\underline{\underline{a}}\ V = \underline{\underline{b}}) \iff ((\neg \underline{\underline{a}})\ Y = \neg\ \underline{\underline{b}}) \qquad (13c)$$

The FORTRAN statement corresponding to (13a) is written as

```
VB = ZV(VA)
```

..30

and the subprogram for ZV becomes, from (13c),

ZV = ZN(ZY(ZN(VA)))

7. Binary Agreement – G

One more binary operator named agreement (G) is defined which will check the two input terms for identity of their logical states in SNS. If they agree with each other, the truth value that it will return is $\underline{\underline{T}}$ and if not, $\underline{\underline{F}}$. This connective will be useful when we want to check whether two forms of expressing an argument are exactly the same, including situations where the terms can be in the $\underline{\underline{D}}$ state. Its truth table is expressible only as a 4 x 4 matrix in SNS, and is shown in Table 7.

Table 7. SNS Truth Table for G

	$\underline{\underline{T}}$	$\underline{\underline{F}}$	$\underline{\underline{D}}$	$\underline{\underline{X}}$
$\underline{\underline{T}}$	$\underline{\underline{T}}$	$\underline{\underline{F}}$	$\underline{\underline{F}}$	$\underline{\underline{F}}$
$\underline{\underline{F}}$	$\underline{\underline{F}}$	$\underline{\underline{T}}$	$\underline{\underline{F}}$	$\underline{\underline{F}}$
$\underline{\underline{D}}$	$\underline{\underline{F}}$	$\underline{\underline{F}}$	$\underline{\underline{T}}$	$\underline{\underline{F}}$
$\underline{\underline{X}}$	$\underline{\underline{F}}$	$\underline{\underline{F}}$	$\underline{\underline{F}}$	$\underline{\underline{T}}$

..31

The logical equation for G is written in the form

$$\underline{\underline{a}}\ G\ \underline{\underline{b}} = \underline{\underline{c}} \tag{14a}$$

and it is defined by the classical equations

$$\begin{aligned} (\text{NOT } (\underline{a}_\alpha \text{ XOR } \underline{b}_\alpha)) \text{ AND } (\text{NOT } (\underline{a}_\beta \text{ XOR } \underline{b}_\beta)) &= \underline{c}_\alpha \\ \text{NOT } \underline{c}_\alpha &= \underline{c}_\beta \end{aligned} \tag{14b}$$

The FORTRAN statement corresponding to (14a) is

```
VC = ZFG(VA,VB)
```

The subprogram for ZFG, is written, using (14b)

```
 AG  =  .NOT.(A(VA).XOR.A(VB))
 AH  =  .NOT.(B(VA).XOR.B(VB))
AVC  =  AG.AND.AH
BVC  =  .NOT.AVC
ZFG  =  STATE(AVC,BVC)
```

V. Reverse Operators

Taking Z to represent a general binary operator, a typical logical equation shall be written as

$$\underline{\underline{a}}\ Z\ \underline{\underline{b}} = \underline{\underline{c}} \tag{15}$$

The equation (15) means that the operation Z acting on the terms $\underline{\underline{a}}$ and $\underline{\underline{b}}$ gives rise to the resultant term $\underline{\underline{c}}$. Suppose there arises a situation wherein we know the resultant term $\underline{\underline{c}}$, and also one of the terms (say $\underline{\underline{a}}$) which is connected by Z to

..32

lead to $\underline{\underline{c}}$; what is the state of $\underline{\underline{b}}$? The operator for this is called a "reverse operator". The reverse operator of Z is represented by the symbol $\overleftarrow{Z}$, and the reverse relation corresponding to Eq.(15) is

$$(\underline{\underline{c}}\ \overleftarrow{Z}\ \underline{\underline{a}} = \underline{\underline{b}}) \equiv (\underline{\underline{a}}\ Z\ \underline{\underline{b}} = \underline{\underline{c}}) \tag{16}$$

1. Reverse And - $\overleftarrow{A}$

We may use the equivalence (16) to obtain the logical equation for $\overleftarrow{A}$ as

$$\underline{\underline{c}}\ \overleftarrow{A}\ \underline{\underline{a}} = \underline{\underline{b}} \tag{17a}$$

and the truth table for $\overleftarrow{A}$, arrived at by analyzing the truth table for the forward A, is as shown in Table 8. The output for a $\underline{\underline{D}}$ input is obtained by using the Boolean addition operator $\oplus$ to add the states given by $\underline{\underline{T}}$ and $\underline{\underline{F}}$ respectively. (The justification for this is given in [7]).

Table 8. Truth Table for $\overleftarrow{A}$, $\overleftarrow{O}$, $\overleftarrow{I}$, $\overleftarrow{J}$

Inputs		$\underline{\underline{c}}\ \overleftarrow{Z}\ \underline{\underline{a}} = \underline{\underline{b}}$ for			
$\underline{\underline{c}}$	$\underline{\underline{a}}$	$\overleftarrow{A}$	$\overleftarrow{O}$	$\overleftarrow{I}$	$\overleftarrow{J}$
$\underline{\underline{T}}$	$\underline{\underline{T}}$	$\underline{\underline{T}}$	$\underline{\underline{D}}$	$\underline{\underline{T}}$	$\underline{\underline{D}}$
$\underline{\underline{T}}$	$\underline{\underline{F}}$	$\underline{\underline{X}}$	$\underline{\underline{T}}$	$\underline{\underline{D}}$	$\underline{\underline{F}}$
$\underline{\underline{F}}$	$\underline{\underline{T}}$	$\underline{\underline{F}}$	$\underline{\underline{X}}$	$\underline{\underline{F}}$	$\underline{\underline{X}}$
$\underline{\underline{F}}$	$\underline{\underline{F}}$	$\underline{\underline{D}}$	$\underline{\underline{F}}$	$\underline{\underline{X}}$	$\underline{\underline{T}}$

..33

From the above table, the classical logical equations for $\underline{b}_\alpha$ and $\underline{b}_\beta$, in terms of $\underline{c}_\alpha$, $\underline{c}_\beta$, $\underline{a}_\alpha$ and $\underline{a}_\beta$ can be worked out to be:

$$\text{NOT}(\underline{c}_\alpha \text{ XOR } \underline{a}_\alpha) = \underline{b}_\alpha \quad ; \quad \underline{c}_\beta = \underline{b}_\beta \tag{17b}$$

The FORTRAN translation of (17a) is

```
VB  =  ZRA(VC,VA)
```

and the subprogram ZRA, corresponding to (17b), is

```
AVB  =  .NOT.(A(VC).XOR.A(VA))
BVB  =  B(VC)
ZRA  =  STATE(AVB,BVB)
```

2. <u>Reverse Or</u> – $\overleftarrow{O}$

Following the same lines as for $\overleftarrow{A}$, the logical equation for $\overleftarrow{O}$ is written as

$$\underline{\underline{c}}\ \overleftarrow{O}\ \underline{\underline{a}} = \underline{\underline{b}} \tag{18a}$$

and its truth table is also shown in Table 8. The classical logical equations representing Eq.(18a) come out to be

$$\underline{b}_\alpha = \underline{c}_\alpha \quad ; \quad \underline{b}_\beta = \text{NOT}(\underline{c}_\beta \text{ XOR } \underline{a}_\beta) \tag{18b}$$

and the subprogram ZRO is, from (18b),

```
AVB  =  A(VC)
BVB  =  .NOT.(B(VC).XOR.B(VA))
ZRO  =  STATE(AVB,BVB)
```

..34

The forward operators A and 0 are commutative in the sense that

$$(\underline{\underline{a}}\ A\ \underline{\underline{b}} = \underline{\underline{c}}) \equiv (\underline{\underline{b}}\ A\ \underline{\underline{a}} = \underline{\underline{c}}) \qquad (19a)$$

$$(\underline{\underline{a}}\ 0\ \underline{\underline{b}} = \underline{\underline{c}}) \equiv (\underline{\underline{b}}\ 0\ \underline{\underline{a}} = \underline{\underline{c}}) \qquad (19b)$$

Hence, (19c) and (19d) hold for the reverse operators A and 0.

$$(\underline{\underline{c}}\ \overleftarrow{A}\ \underline{\underline{a}} = \underline{\underline{b}}) \equiv (\underline{\underline{c}}\ \overleftarrow{A}\ \underline{\underline{b}} = \underline{\underline{a}}) \qquad (19c)$$

$$(\underline{\underline{c}}\ \overleftarrow{0}\ \underline{\underline{a}} = \underline{\underline{b}}) \equiv (\underline{\underline{c}}\ \overleftarrow{0}\ \underline{\underline{b}} = \underline{\underline{a}}) \qquad (19d)$$

<u>3. Reverse Implies</u> – $\overleftarrow{I}$

The logical equation for $\overleftarrow{I}$ is written as

$$\underline{\underline{c}}\ \overleftarrow{I}\ \underline{\underline{a}} = \underline{\underline{b}} \qquad (20a)$$

and since I is not commutative, $\overleftarrow{I}$ is always defined only in the form (20a). From the first equivalence in (10b), it follows that

$$(\underline{\underline{c}}\ \overleftarrow{I}\ \underline{\underline{a}} = \underline{\underline{b}}) \equiv (\underline{\underline{c}}\ \overleftarrow{0}\ \neg\underline{\underline{a}} = \underline{\underline{b}}) \qquad (20b)$$

and hence the truth table for $\overleftarrow{I}$ can be derived from that of $\overleftarrow{0}$ and it is also shown in Table 8. Using (20b) and (18b), the classical logical equation for $\overleftarrow{I}$ can be written as follows:

$$\underline{c}_{\alpha} = \underline{b}_{\alpha} \quad ; \quad \text{NOT}(\underline{c}_{\beta}\ \text{XOR}\ \underline{a}_{\alpha}) = \underline{b}_{\beta} \qquad (20c)$$

..35

The FORTRAN statement corresponding to (20a) for "implies" is

```
VB  =  ZRI(VC,VA)
```

amd the subprogram ZRI is, from (20c), very simply written in terms of the function ZRO as

```
ZRI  =  ZRO(VC,ZN(VA))
```

4. <u>Reverse Implicates</u> – $\overleftarrow{J}$

The logical equation for $\overleftarrow{J}$ can be written as

$$\underline{\underline{c}}\ \overleftarrow{J}\ \underline{\underline{a}} = \underline{\underline{b}} \qquad (21a)$$

Similar to $\overleftarrow{I}$, the following equivalence holds for $\overleftarrow{J}$:

$$(\underline{\underline{c}}\ \overleftarrow{J}\ \underline{\underline{a}} = \underline{\underline{b}}) \qquad (\underline{\underline{c}}\ \overleftarrow{O}\ \underline{\underline{a}} = \neg\underline{\underline{b}}) \qquad (21b)$$

Hence the truth table for $\overleftarrow{J}$ can also be derived from that of $\overleftarrow{O}$ and is shown in Table 8.

Using equations (21b) and (18b), the classical logical equations for $\overleftarrow{J}$ can be written as follows:

$$\underline{b}_{\alpha} = \text{NOT}(\underline{c}_{\beta}\ \text{XOR}\ \underline{a}_{\beta}) \ ; \quad \underline{b}_{\beta} = \underline{c}_{\alpha} \qquad (21c)$$

The FORTRAN statements corresponding to (21a) and (21c) are

```
VB  =  ZRJ(VC,VA)
```

and

```
ZRJ  =  ZN(ZRO(VC,VA))
```

..36

5. Reverse unary operators $\overleftarrow{Y}$ and $\overleftarrow{V}$

As mentioned previously, the logical equation for a unary forward operator Z is written as $\underline{\underline{a}}\ Z = \underline{\underline{b}}$. The correspond-ing reverse relation is written as

$$\underline{\underline{b}}\ \overleftarrow{Z} = \underline{\underline{a}} \tag{22a}$$

Just as the FORTRAN symbol for any unary forward operator is obtained by adding Z in front of its one-letter logical symbol (eg. ZN for N), the FORTRAN symbol for a unary reverse operator, is obtained by adding an R after the symbol for the corresponding forward operator (eg. ZNR for $\overleftarrow{N}$). Hence from Eq.(22), we have the equation

$$(VB = ZZ(VA)) \equiv (VA = ZZR(VB)) \tag{22b}$$

for a general unary operator Z.

In the case of Y and V , it can be shown that

$$(\underline{\underline{a}}\ Y = \underline{\underline{b}}) \equiv (\underline{\underline{b}}\ V = \underline{\underline{a}}) \tag{23a}$$

so that we have

$$\overleftarrow{Y} = V \text{ and } \overleftarrow{V} = Y \tag{23b}$$

Consequently, the FORTRAN statement for $\underline{\underline{b}}\ \overleftarrow{Y} = \underline{\underline{a}}$ is

```
VA = ZYR(VB)
```

in which the subprogram for ZYR takes the simple form

```
ZYR = ZV(VB)
```

..37

Similarly, for $\underline{\underline{b}}\ \overset{\leftarrow}{V} = \underline{\underline{a}}$, the FORTRAN statement is

$$VA = ZVR(VB)$$

and the subprogram for ZVR is

$$ZVR = ZY(VB)$$

In the case of E and N , it is readily seen that the reverses are the same as E and N themselves. Hence, they are not discussed further (see Appendix 1).

6. Multiple operators and their reverses

We can define $\underline{\underline{a1}}\ A\ \underline{\underline{a2}}\ A\ .\ .\ .\ A\ \underline{\underline{an}} = \underline{\underline{c}}$ by the following set of n binary relations, each involving the connective A:

$$\left.\begin{array}{l} \underline{\underline{a1}}\ A\ \underline{\underline{a2}} = \underline{\underline{g2}} \\ \underline{\underline{g2}}\ A\ \underline{\underline{a3}} = \underline{\underline{g3}} \\ \cdots\cdots\cdots \\ \cdots\cdots\cdots \\ \cdots\cdots\cdots \\ \underline{\underline{g(n-1)}}\ A\ \underline{\underline{an}} = \underline{\underline{c}} \end{array}\right\} \qquad (24)$$

The FORTRAN subprogram for this is called ZFAM(VA,N), and no details are given, as the steps are the same as in (24).

In the same way, we can obtain $\underline{\underline{an}}$, given $\underline{\underline{c}}$, $\underline{\underline{a1}}$, $\underline{\underline{a2}}$ ...., and $\underline{\underline{a(n-1)}}$. The corresponding FORTRAN subprogram has the name

..38

ZRAM(VC,VA,J), with J = N-1. Its essential steps are the following:

$$VG = ZFAM(VA,J)$$
$$ZRAM = ZRA(VC,VG)$$

The multiple operators in the forward direction which have been programmed, are A, O, W, U. Only $\overleftarrow{A}$ and $\overleftarrow{O}$ have been programmed in the reverse direction. Both $\overleftarrow{W}$ and $\overleftarrow{U}$ require a much more complicated machinery than SNS , and they have not yet been properly defined (see Appendix 1).

VI. Program NYAYA2 and its Organization

1. General

The subprograms defined in the previous sections are all grouped to form NYAYA2 - the name for the master program. (See Appendix 2)

To solve any practical problem in logic, first its logical graph is drawn as we draw a flow chart for any computational problem, similar to Fig. 1 and Fig.2. Then the logical graph is converted to FORTRAN statements which will form the main program. The main program executed along with NYAYA2 will give the results. Some special considerations are discussed in subsections 2 and 3 below, and they two practical examples are given to illustrate the ease of manipulation of our SNS logic, particularly <u>via</u> FORTRAN programs.

..39

2. Negations of forward and reverse operators

In logic, we often employ negated connectives such as "nand" and "nor". In our implementation, these are implemented in the forms "not and", "not or" etc. Thus,

$$\underline{\underline{a}} \text{ nand } \underline{\underline{b}} = \underline{\underline{c}}\text{, or } \neg(\underline{\underline{a}} \text{ A } \underline{\underline{b}}) = \underline{\underline{c}} \qquad (25a)$$

and

$$\underline{\underline{a}} \text{ nor } \underline{\underline{b}} = \underline{\underline{c}}\text{; or } \neg(\underline{\underline{a}} \text{ O } \underline{\underline{b}}) = \underline{\underline{c}} \qquad (25b)$$

will be written in FORTRAN respectively as:

```
VC = ZN(ZFA(VA,VB))
```

and

```
VC = ZN(ZFO(VA,VB))
```

When it comes to the reverses of such negated operators, the process is best illustrated by the example of "reverse nand", which is the reverse of Eq.(25a). We write it in the form

$$(\underline{\underline{a}} \text{ A } \underline{\underline{b}}) = \neg\underline{\underline{c}} \qquad (26a)$$

so that

$$\neg\underline{\underline{c}} \overleftarrow{\text{A}} \underline{\underline{a}} = \underline{\underline{b}} \qquad (26b)$$

and this has the simple FORTRAN expression

```
VB = ZRA(ZN(VC),VA))
```

In this way, all situations involving negations are dealt with by introducing the function ZN at the appropriate places.

..40

This reduces the number of subprograms necessary in the master program NYAYA2.

3. Procedure for impossible inputs

As already mentioned in II-1, if $\underline{\underline{X}}$ is an input for any connective except for U and G, then the output is also $\underline{\underline{X}}$. Therefore it is not necessary to work out, for instance, the equations in (7a) for obtaining $\underline{\underline{c}}$ (= $\underline{\underline{a}}$ O $\underline{\underline{b}}$), if $\underline{\underline{a}}$ or $\underline{\underline{b}}$ is $\underline{\underline{X}}$, but $\underline{\underline{c}}$ can be put equal to $\underline{\underline{X}}$ straightaway. We call this as the $\underline{\underline{X}}$-priority rule. The FORTRAN subprogram for making an $\underline{\underline{X}}$-check for the inputs is called XCHECK. For any number of inputs VA(I), I = 1, N, the output of XCHECK is •TRUE• if one of them is an SXX , and •FALSE• otherwise. This subprogram for XCHECK(VA,N) has the following structure

```
          Input VA(I), I = 1, N
          DO  3  I = 1, N
          IF  ((VA(I)•EQU•SXX)  GO TO 4
    3     CONTINUE
          XCHECK = •FALSE•
          RETURN
    4     XCHECK = •TRUE•
          END
```

The way in which $\underline{\underline{X}}$-priority rule is implemented using XCHECK is illustrated by the following example of ZFO(VA,VB),

..41

where the output is put straightaway as SXX , if VA or VB is SXX , and the subprogram for O is implemented otherwise.

```
                  VG(1)  =  VA,  VG(2)  =  VB
                  IF (XCHECK(VG,2))  GO TO 4
                  (Subprogram  ZFO  in III - 3)
            4     ZFU  =  SXX
                  RETURN
```

A few practical examples are given below, to illustrate the ease of manipulation of our SNS logic, particularly <u>via</u> FORTRAN programs.

4. Detective problem of Section I-1

We shall consider first the routine procedure of working out the truth table mentioned in I-2. The logical steps are obvious from Fig. 1, and the FORTRAN program of these is obtained as follows:

Input: VP(I),VQ(J),VR(K), with I,J,K = 1,2 for 'T', 'F'

The operative steps in the main program are:

```
            VA  =  ZFO(VP(I),VQ(J))
            VB  =  ZFA(VP(I),ZN(VR(K)))
            VC  =  ZN(ZFS(VQ(J),VR(K))
```

..42

```
      VU1  =  ZFG(STR,VA)
      VU2  =  ZFG(STR,VB)
      VU3  =  ZFG(STR,VC)
   VUU(1)  =  VU1 ; VUU(2) = VU2 ; VUU(3) = VU3
```

Output: VP, VQ, VR, VA, VB, VC, VX, for all I, J, K.

The output of this program executed on the DEC10 computer has been discussed in Section I, Table 1. It is to be noted that no reverse operators are used in this.

If we wish to use the more sophisticated steps in the modification of the same problem, shown in Fig.2, the program is much shorter, but it utilizes reverse operators for four of the steps and the SNS connective W for the fifth. In fact, the problem is almost completely formulated in the syad nyaya system. The essential steps for the FORTRAN program are as follows:

Input VR(I), I = 1, 2 for 'T' , 'F'

Operative steps:

```
      VP1  =  ZRA(STR,ZN(VR(I)))
       VQ  =  ZN(ZRS(STR,VR(I)))
      VP2  =  ZRO(STR,VQ)
       VP  =  ZFW(VP1,VP2)
```

Output: VR, ZN(VR), VQ, VP1, VP2, VP, for each I.

..43

It is to be noted that the second step for VQ is formulated strictly as given in the problem. It is, in fact, equivalent to VQ = ZN(VR(I)), as marked in Fig.2. The output has already been presented in Table 2. and discussed in the Introductory Section I.

It is our hope to develop techniques whereby if the logical graph of a problem is given, it can be fed in as a graph to the computer, and the computer itself can be made to write the FORTRAN program like those mentioned above, and print the results, according to what information is fed in, and what are required to be outputted.

5. Example of a problem from Ref. [6]

A problem, taken from Stoll's well known book "Set theory and Logic" [2] , was discussed in Ref.[6] from this laboratory, and it was shown that, while Stoll has requested only for the proof that one variable ($\underline{\underline{w}}$) is false, a complete analysis made using a FORTRAN program gave unique truth values, namely all false, for 5 of the 6 variables, although the number of equations is only 3, as given below.

The problem is Example 4.3, p.184 of Ref.[2]. Since it has been discussed in detail in Ref.[6] and the logical graph given

..44

therein, only the logical equations, as formulated by us from the graph, and their FORTRAN implementation are given below. The equations, with three variables $\underline{\underline{p}}$, $\underline{\underline{i}}$, $\underline{\underline{c}}$ as inputs are as

$$\left.\begin{aligned}
(\underline{\underline{i}}\ W)\ \overleftarrow{O}\ \underline{\underline{p}} &= \underline{\underline{w}} \\
(\underline{\underline{i}}\ Y)\ \overrightarrow{O}\ \underline{\underline{c}} &= \underline{\underline{s}} \\
(\underline{\underline{s}}\ Y)\ W\ ((\underline{\underline{T}}\ \overrightarrow{A}\ (\underline{\underline{c}}N))N) &= \underline{\underline{u}}
\end{aligned}\right\} \qquad (27)$$

The FORTRAN program is

```
Input:  VP(I), VI(J), VC(K)  ;  I, J, K  =  'T', 'F'
        VW  =  ZRO(ZV(VI(J)),VP(I))
        VS  =  ZRO(ZY(VI(J)),VC(K))
        VU  =  ZFW(ZY(VS),ZN(ZRA(STR,ZN(VC(K)))))
Output:  VP, VI, VC, VS, VU, VW,  for all I, J, K.
```

Since the complete truth table of this problem has not been published, we give the computer output below, in Table 9. It will be seen from the table that one of the outputs, $\underline{\underline{u}}$, $\underline{\underline{w}}$, has the impossible state $\underline{\underline{X}}$, if Eqs.(27) are to be satisfied for seven of the eight possible combinations of the input states of $\underline{\underline{p}}$, $\underline{\underline{i}}$ and $\underline{\underline{c}}$. Hence, only the last case, <u>viz</u> $\underline{\underline{p}} = \underline{\underline{F}}$, $\underline{\underline{i}} = \underline{\underline{F}}$, $\underline{\underline{c}} = \underline{\underline{F}}$ is a permissible one in this problem, and for this one, $\underline{\underline{w}} = \underline{\underline{F}}$, as has been aked to be proved in Stoll's book.

The interesting feature, however, is that in addition to $\underline{\underline{w}}$, <u>all the variables</u> $\underline{\underline{u}}$, $\underline{\underline{p}}$, $\underline{\underline{i}}$, $\underline{\underline{c}}$ <u>also can take up only one state</u>,

..45

TABLE 9: TRUTH TABLE FOR THE PROBLEM TAKEN FROM STOLL'S BOOK

P	I	C	S	U	W
T	T	T	D	X	D
T	T	F	T	X	D
T	F	T	D	X	X
T	F	F	D	F	X
F	T	T	D	X	D
F	T	F	T	X	D
F	F	T	D	X	F
F	F	F	D	F	F

..46

namely $\underline{\underline{F}}$, and only $\underline{\underline{s}}$ can be either $\underline{\underline{T}}$ or $\underline{\underline{F}}$. It is very difficult to show this by standard techniques. It will be noticed how simple it is to treat such complicated problems in the syad-nyaya system, and how very little effort is required to be put in. The techniques of solution becomes routine and is capable of being handled by a computer.

In fact, we may say that the adoption of the technique developed by us, which employs a vector-matrix representation of the logical content of connected statements, is very similar to the use of Cartesian coordinate geometry to solve problems, in practice, in physics and engineering (instead of the axiomatic Euclidean geometry). In a similar way, we build in the axiomatic principles of logic into a set of formulae for logical states, connectives, comparison of terms etc, and then these formulae become very simple and easy to use for solving practical problems.

Acknowledgements

We are very grateful to Mr. R.E.C. Johnson who constructed a good part of the electronic analog computer for logic, based on the principles discussed in this paper, and which helped us very much in developing the FORTRAN programs. Another paper is being written on this aspect. We are grateful to the University Grants Commission, India, for partial financial assistence.

..47

References

1. Suppes, P. "Introduction to Logic", 1957, New York, Van Nostrant

2. Stoll, R.R. "Set Theory and Logic", 1961, San Francisco, W.H. Freeman

3. Copi, I.M. "Symbolic Logic", 1979, New York, Macmillan

4. Halmos, P.R. "Lectures on Boolean Algebra", 1974, New York, Springer-Verlag

5. Ramachandran, G.N. "Syad-Nyaya-System Logic - Essential ideas connected with propositional calculus" - Matphil Reports No.1, 1979

6. Ramachandran, G.N. "Computerization of Logic", Golden Jubilee Commemoration Volume, Natl. Acad. Sci. India, Allahabad, 1980 (in press) (Matphil Reports No.8)

7. Ramachandran, G.N. "Principles of Practical Logic", (under preparation)

8. Ramachandran, G.N., Johnson, R.E.C. and Thanaraj, T.A., "ESNY - an Analog Computer for Logic", (submitted for publication).

..48

Appendix 1 - List of subprograms available in NYAYA2

Logical symbol	FORTRAN symbol	Nature of Function
	1. Basic functions	
$\underline{a}_{\alpha}$, $\underline{a}_{\beta}$	A(VA),B(VA)	Finds the α and β components of
$\underline{a}(\alpha , \beta)$	STATE(AVA,BVA)	Finds SNS truth value of VA, giv α and β components
$\underline{\underline{a}}(\underline{\underline{i}}) = \underline{\underline{X}}$,	XCHECK(VA,N)	Checks for the state $\underline{\underline{X}}$ for any of the VA(I)'s, I = 1 to N
	2. Forward and reverse unary connectives	
E ; $\overleftarrow{E}$	ZE(VA) ; ZER(VA)*	equal to, and its reverse
N ; $\overleftarrow{N}$	ZN(VA) ; ZNR(VA)*	negation of, and its reverse
Y ; $\overleftarrow{Y}$	ZY(VA) ; ZYR(VA)$^+$	implies, and its reverse
V ; $\overleftarrow{V}$	ZV(VA) ; ZVR(VA)$^+$	implicates, and its reverse

*ZER = ZE, ZNR = ZN, $^+$ZYR = ZV, ZVR = ZY

Logical symbol	FORTRAN symbol	Nature of Function
	3. Forward and reverse binary connectives	
A ; $\overleftarrow{A}$	ZFA(VA,VB) ; ZRA(VA,VB)	and, and reverse
O ; $\overleftarrow{O}$	ZFO(VA,VB) ; ZRO(VA,VB)	or, and reverse
I ; $\overleftarrow{I}$	ZFI(VA,VB) ; ZRI(VA,VB)	implies, and reverse
J ; $\overleftarrow{J}$	ZFJ(VA,VB) ; ZRJ(VA,VB)	implicates, and reverse
S	ZFS(VA,VB) ; ZRS = ZFS	equivalent, same
W	ZFW(VA,VB) (Reverses of W, U and G have not yet been defined	with
U	ZFU(VA,VB)	upon
G	ZFG(VA,VB)	agree

..49

4. Forward and reverse multiple connectives

A	ZFAM(VA,N)	multiple and,
$\overleftarrow{A}$	ZRAM(VC,VA,J)	its reverse, J = N-1
O	ZFOM(VA,N)	multiple or,
$\overleftarrow{O}$	ZROM(VC,VA,J)	its reverse, J = N-1
W	ZFWM(VA,N)	multiple with
U	ZFUM(VA,N)	multiple upon

Appendix 2 - Listing of NYAYA2 programs

```
      LOGICAL FUNCTION A(VA)
C     THIS SUBPROGRAM COMPUTES THE ALPHA COMPONENT OF THE VARIABLE
      IMPLICIT LOGICAL (A-B,X)
      DATA STR,SFL,SDF,SXX/'T','F','D','X'/
      IF ((VA .EQ. STR) .OR. (VA .EQ. SDF)) GO TO 3
      A = .FALSE.
      RETURN
3     A = .TRUE.
      END
      LOGICAL FUNCTION B(VA)
C     THIS SUBPROGRAM COMPUTES THE BETA COMPONENT OF THE VARIABLE VA
      IMPLICIT LOGICAL (A-B,X)
      DATA STR,SFL,SDF,SXX/'T','F','D','X'/
      IF ((VA .EQ. STR) .OR. (VA .EQ. SXX)) GO TO 3
      B = .TRUE.;RETURN
3     B = .FALSE.
      END
      LOGICAL FUNCTION STATE(AVA,BVA)
C     THIS SUBPROGRAM COMPUTES THE TRUTH VALUE OF THE VARIABLE VA,GI
C     ITS COMPONENTS.
      IMPLICIT LOGICAL (A-B,X)
      DATA STR,SFL,SDF,SXX/'T','F','D','X'/
      IF (AVA) GO TO 3
      IF (BVA) GO TO 4
      STATE=SXX;RETURN
4     STATE=SFL;RETURN
3     IF (BVA) GO TO 5
      STATE=STR;RETURN
5     STATE=SDF
      END
```

..51

```
LOGICAL FUNCTION XCHECK(VA,N)
THIS SUBPROGRAM CHECKS WHETHER ANY ONE OF VA IS X AND RETURNS
  IF SO ANL FALSE IF NOT
DATA STR,SFL,SDF,SXX/'T','F','D','X'/
DIMENSION VA(N)
DO 3 I=1,N
IF (VA(I) .EQ. SXX) GO TO 4
XCHECK=.FALSE.;RETURN
XCHECK = .TRUE.
END
LOGICAL FUNCTION ZE(VA)
THIS SUBPROGRAM COMPUTES THE TRUTH VALUE OF THE UNARY OPER
EQUAL TO ACTING ON THE INPUT VA
IMPLICIT LOGICAL (A-B,X)
DIMENSION VAA(1)
VAA(1)=VA
IF (XCHECK(VAA,1)) GO TO 3
ZE=VA;RETURN
ZE=SXX
END
LOGICAL FUNCTION ZER(VA)
THIS SUBPROGRAM COMPUTES THE TRUTH VALUE OF THE OPERATION REV
EQUAL TO ON THE INPUT VA.
IMPLICIT LOGICAL (A-B,X)
DATA STR,SFL,SDF,SXX/'T','F','D','X'/
DIMENSION VG(1)
VG(1)=VA
IF (XCHECK(VG,1)) GO TO 4;ZER=ZE(VA);RETURN
ZER=SXX
END
```

..52

```
LOGICAL FUNCTION ZN(VA)
THIS SUBPROGRAM COMPUTES THE TRUTH VALUE OF THE OPERATION UNAR
NEGATION ACTING ON THE INPUT VA.
  IMPLICIT LOGICAL (A-B,X)
  DIMENSION VAA(1)
  DATA STR,SFL,SDF,SXX/'T','F','D','X'/
  VAA(1)=VA
  IF (XCHECK(VAA,1)) GO TO 3
  ZN=STATE(B(VA),A(VA));RETURN
  ZN=SXX
  END
LOGICAL FUNCTION ZNR(VA)
THIS SUBPROGRAM COMPUTES THE TRUTH VALUE OF THE OPERATION UNAR
REVERSE NEGATION ACTING ON THE INPUT VA.
IMPLICIT LOGICAL (A-B,X)
DATA STR,SFL,SDF,SXX/'T','F','D','X'/
DIMENSION VG(1)
VG(1)=VA
IF (XCHECK(VG,1)) GO TO 4
ZNR=ZN(VA);RETURN
ZNR=SXX
END
LOGICAL FUNCTION ZY(VA)
THIS SUBPROGRAM COMPUTES THE TRUTH VALUE OF THE OPERATION UNAR
ACTING ON THE INPUT VA.
IMPLICIT LOGICAL (A-B,X)
DIMENSION VAA(1)
DATA STR,SFL,SDF,SXX/'T','F','D','X'/
VAA(1)=VA
IF (XCHECK(VAA,1)) GO TO 3
ZY=STATE(.TRUE.,B(VA))
RETURN
ZY=SXX
END
```

..53

```
 LOGICAL FUNCTION ZYR(VA)
THIS SUBPROGRAM COMPUTES THE TRUTH VALUE OF THE OPERATION UNAR
REVERSE IMPLIES ACTING ON THE INPUT VA.
IMPLICIT LOGICAL (A-B,X)
DATA STR,SFL,SDF,SXX/'T','F','D','X'/
DIMENSION VG(1)
VG(1)=VA;IF (XCHECK(VG,1)) GO TO 4
ZYR=ZV(VA);RETURN
ZYR=SXX
END

LOGICAL FUNCTION ZV(VA)
THIS SUBPROGRAM COMPUTES THE TRUTH VALUE OF THE OPERATION UNAR
IMPLICATES ON THE INPUT VA.
IMPLICIT LOGICAL (A-B,X)
DIMENSION VAA(1)
DATA STR,SFL,SDF,SXX/'T','F','D','X'/
VAA(1)=VA
IF (XCHECK(VAA,1)) GO TO 3
ZV=STATE(A(VA),.TRUE.);RETURN
ZV=SXX
 END

LOGICAL FUNCTION ZVR(VA)
THIS SUBPROGRAM COMPUTES THE TRUTH VALUE OF THE OPERATION UNAR
REVERSE IMPLICATES ACTING ON THE INPUT VA.
IMPLICIT LOGICAL (A-B,X)
DATA STR,SFL,SDF,SXX/'T','F','D','X'/
DIMENSION VG(1)
VG(1)=VA;IF (XCHECK(VA,1)) GO TO 4
ZVR=ZY(VA);RETURN
ZVR=SXX
END
```

..54

```
LOGICAL FUNCTION ZFA(VA,VB)
THIS SUBPROGRAM COMPUTES THE TRUTH VALUE OF THE OPERATION FORWARD
AND ON VA AND VB
IMPLICIT LOGICAL (A-B,X)
DIMENSION VAA(2)
DATA STR,SFL,SDF,SXX/'T','F','D','X'/
VAA(1)=VA;VAA(2)=VB
IF (XCHECK(VAA,2)) GO TO 3
ZFA=STATE(A(VA) ·AND· A(VB),B(VA) ·OR· B(VB));RETURN
ZFA=SXX
END

 LOGICAL FUNCTION ZRA(VC,VA)
THIS SUBPROGRAM COMPUTES THE TRUTH VALUE OF THE OPERATOR REVERSE
AND ACTING ON THE VARIABLES VC AND VA.
IMPLICIT LOGICAL (A-B,G-H,X)
DIMENSION VAAA(1),VAA(2)
 DATA STR,SFL,SDF,SXX/'T','F','D','X'/
VAA(1)=VC;VAA(2)=VA
IF (XCHECK(VAA,2)) GO TO 3
VAAA(1)=VA
          G=A(VC) ·AND· A(VA);H=B(VC) ·AND· B(VA)
          ZRA=STATE(G ·OR· H,B(VC));RETURN
ZRA=SXX
END

LOGICAL FUNCTION ZFO(VA,VB)
  THIS SUBPROGRAM COMPUTES THE TRUTH VALUE OF THE OPERATION
  BINARY OR ACTING ON THE INPUTS VA AND VB.
IMPLICIT LOGICAL (A-B,X)
DIMENSION VAA(2)
DATA STR,SFL,SDF,SXX/'T','F','D','X'/
VAA(1)=VA;VAA(2)=VB
IF (XCHECK(VAA,2)) GO TO 3
```

..55

```
  ZFO=STATE(A(VA) •OR• A(VB),B(VA) •AND• B(VB));RETURN
 ZFO=SXX
 END
 LOGICAL FUNCTION ZRO(VC,VA)
 THIS SUBPROGRAM COMPUTES THE TRUTH VALUE OF THE OPERATOR REVERSE
 OR ACTING ON THE VARIABLES VC AND VA.
 IMPLICIT LOGICAL (A-B,X,G-H)
 DIMENSION VAAA(1),VAA(2)
 DATA STR,SFL,SDF,SXX/'T','F','D','X'/
 VAA(1)=VC;VAA(2)=VA
 IF (XCHECK(VAA,2)) GO TO 3
 VAAA(1)=VA
   G=A(VC)  AND  A(VA);H=B(VC) •AND• B(VA)
   ZRO=STATE(A(VC),G •OR• H);RETURN
 ZRO=SXX
 END
 LOGICAL FUNCTION ZFI(VA,VB)
 THIS SUBPROGRAM COMPUTES THE TRUTH VALUE OF THE OPERATION
 BINARY IMPLIES ACTING ON THE INPUTS VA AND VB.
 IMPLICIT LOGICAL (A-B,X)
 DIMENSION VAA(2)
 DATA STR,SFL,SDF,SXX/'T','F','D','X'/
 VAA(1)=VA;VAA(2)=VB
 IF (XCHECK(VAA,2)) GO TO 3
 ZFI=ZFO(ZN(VA),VB);RETURN
 ZFI=SXX
 END
   LOGICAL FUNCTION ZRI(VC,VA)
 THIS SUBPROGRAM COMPUTES THE TRUTH VALUE OF THE OPERATOR BINARY
 REVERSE IMPLICATION ACTING ON THE VARIABLES VC AND VA UNDER THE
 CONDITION THAT VA IMPLIES VB GIVES VC,
  IMPLICIT LOGICAL (A-B,X)
 DATA STR,SFL,SDF,SXX/'T','F','D','X'/
```

..56

```
      DIMENSION VAA(2)
      VAA(1)=VC,VAA(2)=VA
      IF  (XCHECK(VAA,2)) GO TO 3
      ZRI=ZRO(VC,ZN(VA));RETURN
 3     ZRI=SXX
      END

      LOGICAL FUNCTION ZFJ(VA,VB)
C     THIS SUBPROGRAM COMPUTES THE TRUTH VALUE OF THE OPERATOR BINARY
C     IMPLICATES ACTING ON THE VARIABLES VA AND VB UNDER THE CONDITION
C     THAT VA IMPLICATES VB.
      IMPLICIT LOGICAL (A-B,X)
      DATA STR,SFL,SDF,SXX/'T','F','D','X'/
      DIMENSION VAA(2)
      VAA(1)=VA;VAA(2)=ZN(VB)
      IF (XCHECK(VAA,2)) GO TO 3
      ZFJ=ZFOM(VAA,2)
      RETURN
 3    ZFJ=SXX
      END

      LOGICAL FUNCTION ZRJ(VC,VA)
C     THIS SUBPROGRAM COMPUTES THE TRUTH VALUE OF THE OPERATOR BINARY
C     REVERSE IMPLICATES ACTING ON THE VARIABLES VA AND VB UNDER THE
C     CONDITION THAT VA IMPLICATES VB.
      IMPLICIT LOGICAL (A-B,X)
      DIMENSION VAA(2)
      DATA STR,SFL,SDF,SXX/'T','F','D','X'/
      VAA(1)=VC;VAA(2)=VA
      IF (XCHECK(VAA,2)) GO TO 3
      ZRJ=ZN(ZRO(VC,VA));RETURN
 3    ZRJ=SXX
      END
```

..57

```
LOGICAL FUNCTION ZFS(VA,VB)
THIS SUBPROGRAM COMPUTES THE TRUTH VALUE OF OPERATOR BINARY SIMIL
ACTING ON THE VARIABLES VA AND VB
IMPLICIT LOGICAL (A-B,X,G-H)
DIMENSION VAA(2)
DATA STR,SFL,SDF,SXX/'T','F','D','X'/
VAA(1)=VA;VAA(2)=VB
IF (XCHECK(VAA,2)) GO TO 3
VG=ZFG(VA,VB)
G1=A(VA) .AND. B(VA);G2=A(VB) .AND. B(VB)
H=G1 .OR. G2
AVC=A(VG) .OR. H;BVC=B(VG) .OR. H
ZFS=STATE(AVC,BVC);RETURN
ZFS=SXX
END
LOGICAL FUNCTION ZRS(VC,VA)
THIS SUBPROGRAM COMPUTES THE TRUTH VALUE OF THE OPERATOR BINARY
REVERSE SIMILAR ACTING ON THE VARIABLES VC AND VA
IMPLICIT LOGICAL (A-B,X)
DIMENSION VAA(2)
 DATA STR,SFL,SDF,SXX/'T','F','D','X'/
VAA(1)=VC;VAA(2)=VA
IF (XCHECK(VAA,2)) GO TO 3
ZRS=ZFS(VC,VA);RETURN
ZRS=SXX
END
LOGICAL FUNCTION ZFW(VA,VB)
 THIS SUBPROGRAM COMPUTES THE TRUTH VALUE OF THE OPERATOR FORWARD
WITH ON THE VARIABLES VA AND VB.
IMPLICIT LOGICAL (A-B,X)
DIMENSION VAA(2)
DATA STR,SFL,SDF,SXX/'T','F','D','X'/
VAA(1)=VA;VAA(2)=VB
```

..58

```
IF (XCHECK(VAA,2)) GO TO 3
  ZFW=STATE(A(VA) •AND• A(VB),B(VA) •AND• B(VB));RETURN
ZFW=SXX
END

  LOGICAL FUNCTION ZFU(VA,VB)
 THIS SUBPROGRAM COMPUTES THE TRUTH VALUE OF THE OPERATOR FORWARD
UPON ACTING ON THE VARIABLES VA AND VB.
IMPLICIT LOGICAL (A-B,X)
DIMENSION VAA(2)
DATA STR,SFL,SDF,SXX/'T','F','D','X'/
VAA(1)=VA;VAA(2)=VB
ZFU=ZFUM(VAA,2);RETURN
END

 LOGICAL FUNCTION ZFG(VA,VB)
THIS SUBPROGRAM COMPUTES THE TRUTH VALUE OF THE OPERATOR
BINARY AGREEMENT ACTING ON THE VARIABLES VA AND VB.
IMPLICIT LOGICAL (A-B,X)
DIMENSION VAA(2)
DATA STR,SFL,SDF,SXX/'T','F','D','X'/
AG=A(VA) •XOR• A(VB);BG=B(VA) •XOR• B(VB)
AH= •NOT• AG •AND• •NOT• BG
ZFG=STATE(AH,•NOT• AH);RETURN
END

LOGICAL FUNCTION ZFAM(VA,N)
THIS SUBPROGRAM COMPUTES THE TRUTH VALUE OF THE
FORWARD AND OPERATION ACTING ON THE VARIABLES VA(1) TO VA(N).
IMPLICIT LOGICAL (A-B,X)
DIMENSION VA(N)
DATA STR,SFL,SDF,SXX/'T','F','D','X'/
IF (XCHECK(VA,N)) GO TO 5
IF (N •GT• 2) GO TO 3
 ZFAM=ZFA(VA(1),VA(2))
```

..59

```
RETURN
AG=.TRUE.;BG=.FALSE.
DO 4 J=1,N
AG=AG .AND. A(VA(J))
BG=BG .OR. B(VA(J))
ZFAM=STATE(AG,BG);RETURN
ZFAM=SXX
END

LOGICAL FUNCTION ZRAM(VC,VA,J)
THIS SUBPROGRAM COMPUTES THE TRUTH VALUE OF THE OPERATOR
REVERSE AND ACTING ON THE  VC AND THE RESULT OF THE OPERATION AND
ACTING ON THE VARIABLES VA(1) TO VA(J).   THIS SUBPROGRAM TAKES
CARE OF THE BINARY CASE ALSO.
IMPLICIT LOGICAL (A-B,X,G-H)
DIMENSION VA(J),VAA(10)
DATA STR,SFL,SDF,SXX/'T','F','D','X'/
VAA(1)=VC; DO 3 J1=1,j
VAA(J1+1)=VA(J1)
IF (XCHECK(VAA,J+1)) GO TO 4
VA1=VA(1)
IF (J .GT. 1) VA1=ZFA(VA,J)
G=A(VC) .AND. A(VA1);H=B(VC) .AND. B(VA1)
 ZRAM=STATE(G .OR. H,B(VC));RETURN
  ZRAM=SXX
END

LOGICAL FUNCTION ZFOM(VA,N)
THIS SUBPROGRAM COMPUTES THE TRUTH VALUE OF THE OPERATION OI
ACTING ON THE VARIABLES VA(1) TO VA(N). THIS TAKES CARE
OF THE BINARY CASE ALSO
IMPLICIT LOGICAL (A-B,X,G-H)
DIMENSION VA(N)
DATA STR,SFL,SDF,SXX/'T','F','D','X'/
IF (XCHECK(VA,N)) GO TO 5
```

..60

```
 IF (N .GT. 2) GO TO 3
  ZFOM=ZFO(VA(1),VA(2))
 RETURN
 AG=.FALSE.;BG=.TRUE.
 DO 4 J=1,N
 AG=AG .OR. A(VA(J))
 BG=BG .AND. B(VA(J))
 ZFOM=STATE(AG,BG);RETURN
 ZFOM=SXX
 END
 LOGICAL FUNCTION ZROM(VC,VA,J)
 THIS SUBPROGRAM COMPUTES THE TRUTH VALUE OF THE OPERATION REVERSE
 OR ACTING ON THE VARIABLES VC AND THE RESULT OF THE OPERATION OR
 ACTING ONE THE VARIABLE VA(1) TO VA(J).  THIS SUBPROGRAM TAKES CARE
 OF THE BINARY OPERATOR ALSO.
 IMPLICIT LOGICAL (A-B,X,G-H)
 DIMENSION VA(J),VAA(10)
 DATA STR,SFL,SDF,SXX/'T','F','D','X'/
 VAA(1)=VC;DO 3 J1=1,J
 VAA(J1)=VA(J1)
 IF (XCHECK(VAA,J+1)) GO TO 4
 VA1=VA(1)
 IF (J .GT. 1) VA1=ZFOM(VA,J)
 G=A(VC) .AND. A(VA1);H=B(VC) .AND. B(VA1)
 ZROM=STATE(A(VC),G .OR. H);RETURN
 ZROM=SXX
 END

 LOGICAL FUNCTION ZFUM(VA,N)
 THIS SUBPROGRAM COMPUTES THE TRUTH VALUE OF THE OPERATION UPON
 ACTING ON THE VARIABLES VA(1)= TO VA(N).  THIS SUBPROGRAM TAKES CARE
 OF THE BINARY CASE ALSO.
 IMPLICIT LOGICAL (A-B,X)
 DIMENSION VA(N)
```

..61

```
      DATA STR,SFL,SDF,SXX/'T','F','D','X'/
      IF (N .GT. 2) GO TO 3
        ZFUM = ZFU(VA(1),VA(2))
      RETURN
3     AG=.FALSE.;BG=.FALSE.
      DO 7 J=1,N
      AG=AG .OR. A(VA(J))
7     BG=BG .OR. B(VA(J))
      ZFUM=STATE(AG,BG)
      RETURN
      END

      LOGICAL FUNCTION ZFWM(VA,N)
C     THIS SUBPROGRAM COMPUTES THE TRUTH VALUE OF THE OPERATION WITH
C     ACTING ON THE INPUTS VA(1) TO VA(N). THIS SUBPROGRAM TAKES CARE
C     THE BINARY CASE ALSO.
      IMPLICIT LOGICAL (A-B,X)
      DIMENSION VA(N)
      DATA STR,SFL,SDF,SXX/'T','F','D','X'/
      IF (XCHECK(VA,N)) GO TO 7
       IF (N .GT. 2) GO TO 3
      ZFWM = ZFW(VA(1),VA(2))
      RETURN
3     AG=.TRUE.;BG=.TRUE.
      DO 4 J=1,N
      AG=AG .AND. A(VA(I))
4     BG=BG .AND. B(VA(I))
      ZFWM=STATE(AG,BG);RETURN
7     ZFWM=SXX
       END
```

E L E C T R O N I C   S Y Ā D   N Y Ā Y A   Y A N T R A*

(E S N Y - 2)

(Analog Computer for Sentential Logic)

G.N. Ramachandran, R.E.C. Johnson, T.A. Thanaraj

Mathematical Bhilosophy Group
Indian Institute of Science
BANGALORE-560 012

*MATPHIL Reports No. 13, October 1980.

CONTENTS

Contd..

ELECTRONIC SYĀD-NYĀYA YANTRA
(E S N Y)
(Analog Computer for Sentential Logic)

I. Introduction

This report gives a description of the logic machine which was built for implementing electronically the various operations for representing the connectives of Sentential Logic and has been named as the "Electronic Syād Nyāya Yantra" (ESNY). The name comes from the use of an extension of classical two-valued logic (having two states T and F) by including, in addition, two other states, viz. the doubtful state (D) and the inpossible state (X) [1]. The word Syād (Sanskrit) = may be, was used by the Jaina philosophers of the first millenium B.C. in India who gave the name "Syād-Vāda" (dilectics of doubt) for their system of epistemolog and logic. Nyāya is the Sanskrit word for "logic" and Yantra in Sanskrit means "machine". Thus ESNY, when translated, will

..2

read "Electronic may-be-logic machine".

This machine can be briefly described as a logic computer, in the sense that it can be used to evaluate the truth value of a logically connected sentence, when the truth values of the input data are put into the machine. The machine uses TTL-gates for implementing all the operations. In this sense, we have designed our computer completely on the basis of classical logic, which is well-known to be implementable by means of the elements - EQU, NOT, AND, OR, XOR. In fact, a question might arise as to how four-level logical operations can be implemented using only TTL-gates, which are essentially binary gates. This is made possible by the use of two electrical lines (each having the states 0 and 1) to represent a variable, or constant, describing the logical state of the corresponding term. In effect, therefore, the logic is based on two-component Boolean vectors. We have given this modification the name syād-nyāya-system (SNS) logic. The principle of this is

..3

briefly discussed in [2]. A general account of the way of converting the principles of SNS logic so as to be capable of being implemented in terms of classical logic gates is discussed in [2], and [3] gives the FORTRAN program that has been written for implementing these logical principles on a digital computer.

Actually, the development of the computer programs was later than the development of the logic machine described here. However, for various reasons, the latter was written first and therefore we shall try to make use of the logical principles governing the application and use of SNS logic, as discussed in the previous papers. In this report, we shall emphasize mainly the electronic aspects of the implementation and also practical details, such as the way in which the various circuits are constructed, the arrangement of the modules in the panel, and the techniques of putting in problems and taking

..4

out data from the computer, etc.

Stated briefly, the input to any operator-module consists of two components, each of which is a classical Boolean variable, so that the four states are represented by (true) $\underline{\underline{T}} = (1 \quad 0)$, (false) $\underline{\underline{F}} = (0 \quad 1)$, (doubtful) $\underline{\underline{D}} = (1$ and (impossible) $\underline{\underline{X}} = (0 \quad 0)$. The input data can be read out from glowing lamps, the state being determined by the colour of the lamp (green for $\underline{\underline{T}}$, red for $\underline{\underline{F}}$, with both green and red shining when it is $\underline{\underline{D}}$ and both being dark for the state $\underline{\underline{X}}$). Actually, the state $\underline{\underline{X}}$ is never used as an input, and occurs only as an output of an intermediate step, in which case it is an indication of the existence of some contradiction in the data, or the problem, that is fed in. Therefore ESNY responds to such a situation, when $\underline{\underline{X}}$ comes as an output, by sounding an alarm, corresponding to the result "impossible", or "contradictory". It also indicates in which

..5

module the logical inconsistency occurs, by flashing a yellow lamp in the relevant module, in addition to sounding the alarm (see Sec. IV-2).

Briefly, the method of utilizing ESNY is as follows. For any problem, the logical interconnections between the different terms, or sentences, via connectives such as and, or, not, equ, etc., are first drawn in the form of a graph as indicated in Sec. VIII. The logical interconnections correspondi to those in the graph are then made in the computer by means of electrical cables which connect one module to another (See Sec. III-1 for details). Having thus loaded the problem on the analog computer, the results are obtained by setting each of the inputs so as to take up the relevant state - $\underline{\underline{T}}$, $\underline{\underline{F}}$ or $\underline{\underline{D}}$, as the case may be. For each combination of states of the inputs, the output is taken to the output module (see Sec. II), and the state of the output is read out from the lamps in the output module. Thus the machine is very versatile and any required

..6

test can be carried out, or any particular question that is asked can be answered, with regard to the wired problem. For instance, we have a module for the connective called "with", which checks whether two statements are contradictory, or consistent, and gives an output $\underline{\underline{X}}$ in the former case. In the same way, there is a module "agree", which shows a green signal only if both the inputs are identical among the four SNS truth values. This can be used to check for a similar situation in a problem. These details are given in Sec. IV-2,3.

In the succeeding sections, we shall deal essentially with the electronic and electrical aspects of the instrument and we shall describe the principle and construction of the various modules and the way in which these are interconnected for implementing a problem on the computer. We shall follow this up, by giving some examples of real problems which can be tested on the machine and for which it gives very rapid answers with practically no thinking effort involved in this. It must

..7

be emphasized that the construction of this analog computer is based almost completely on logic gates which are well-known in computer circuits and on principles based on classical two-valued logic. The relationship between classical logic and the extended SNS logic is discussed in full detail in a book being written by one of the authors [4] , and therefore we shall not give much theoretical details in this paper, but mention only special principles of this nature which are employed in the design of the computer.

In the succeeding section, we shall describe the details of the circuits and the layouts of the various modules in ESNY. The machine has a front panel made up of four parts, labelled Panel I, Panel II, Panel III and Panel IV. Layout of these are shown in Figs. 1, 2, 3 and 4 respectively. The names given to individual modules in these panels will be referred to in the sections that follow, which deal with the construction of the machine (both its electronic circuits, and mechani-

..8

cal layout), and the description of its operation.

Figures. 1, 2, 3, 4. Layout of the various modules of Panels I, II, III, IV, respectively, of ESNY.

It should be mentioned that this report deals essentially with ESNY-2, the second stage of development of ESNY. In the initial stage only the designs of the input and output modules, including intermediate inputs, the binary connectives "and", "or" and "implies" (I) and the unary connectives "Equ", "not", "implies" (Y), and combinations of these were developed. We then assured ourselves that the analog computer had all the required facilities for checking simple logical arguments in sentential logic and that it could then be developed for implementing the SNS operators also. One module for each of "with" (W) and "upon" (U) were also tried out, including the alarm signals at the first stage. The ESNY-1 had only two panels and some fifteen modules which was adequate for initial tests. However,

..9

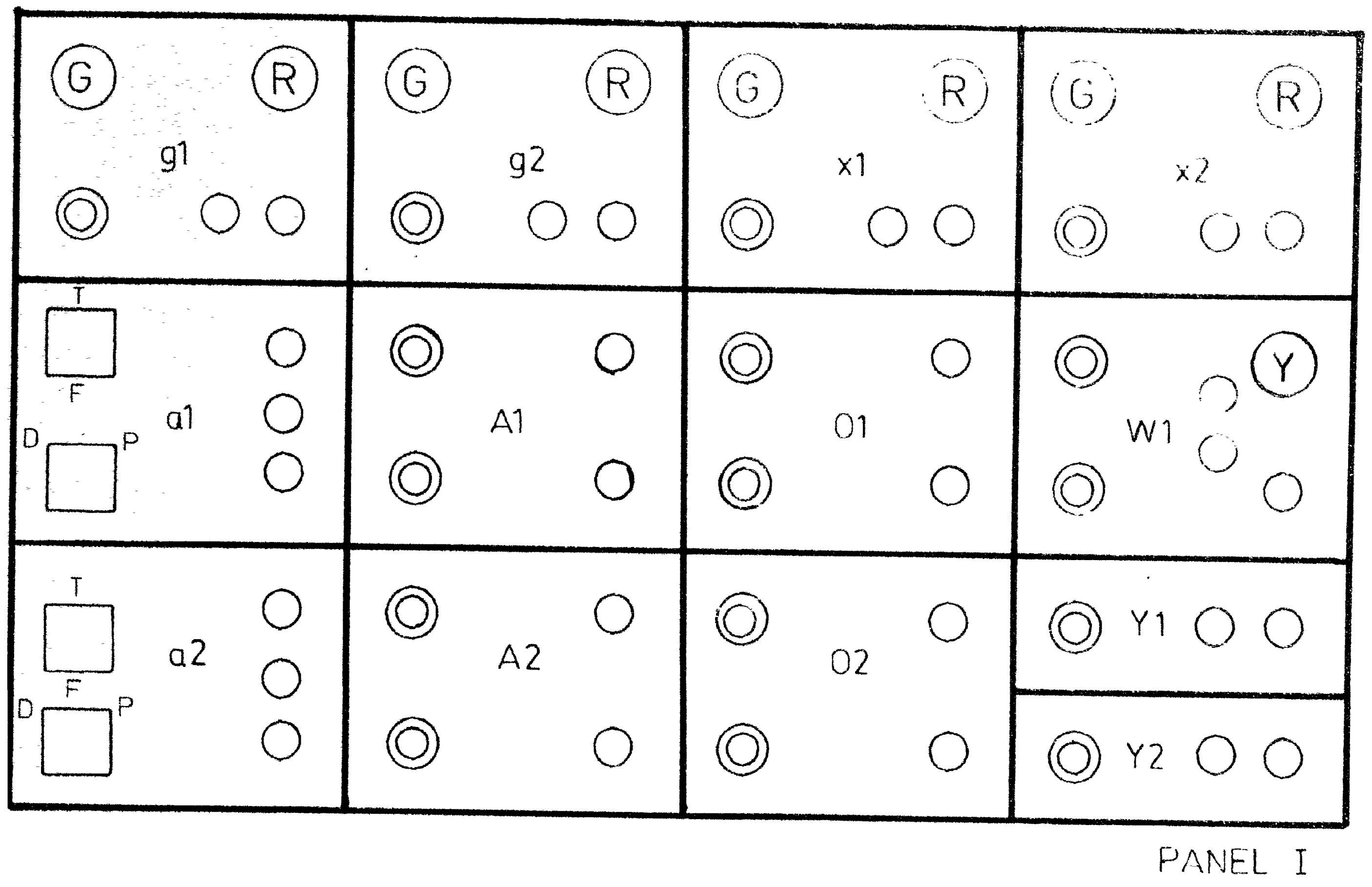


Fig. 1

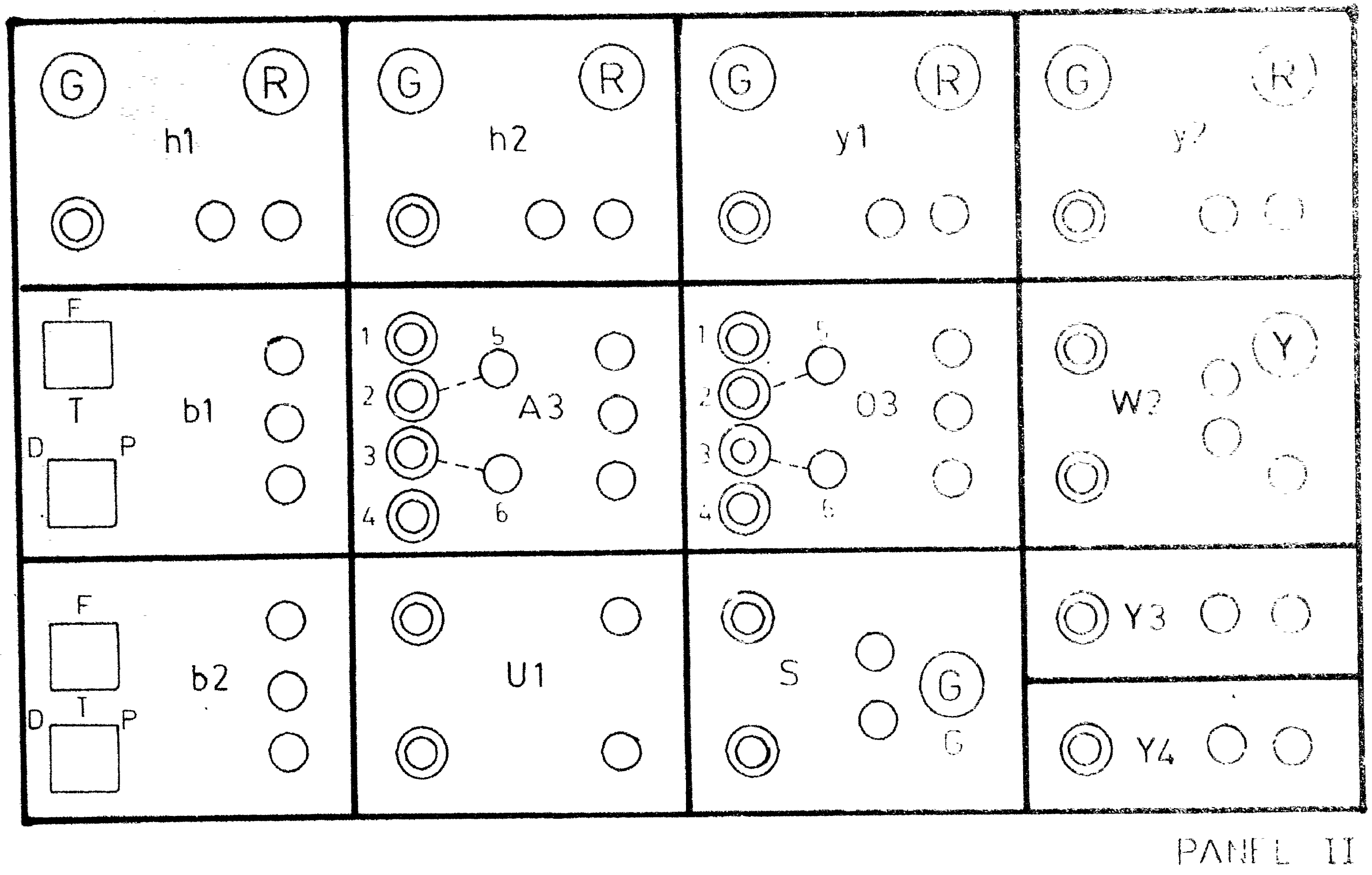
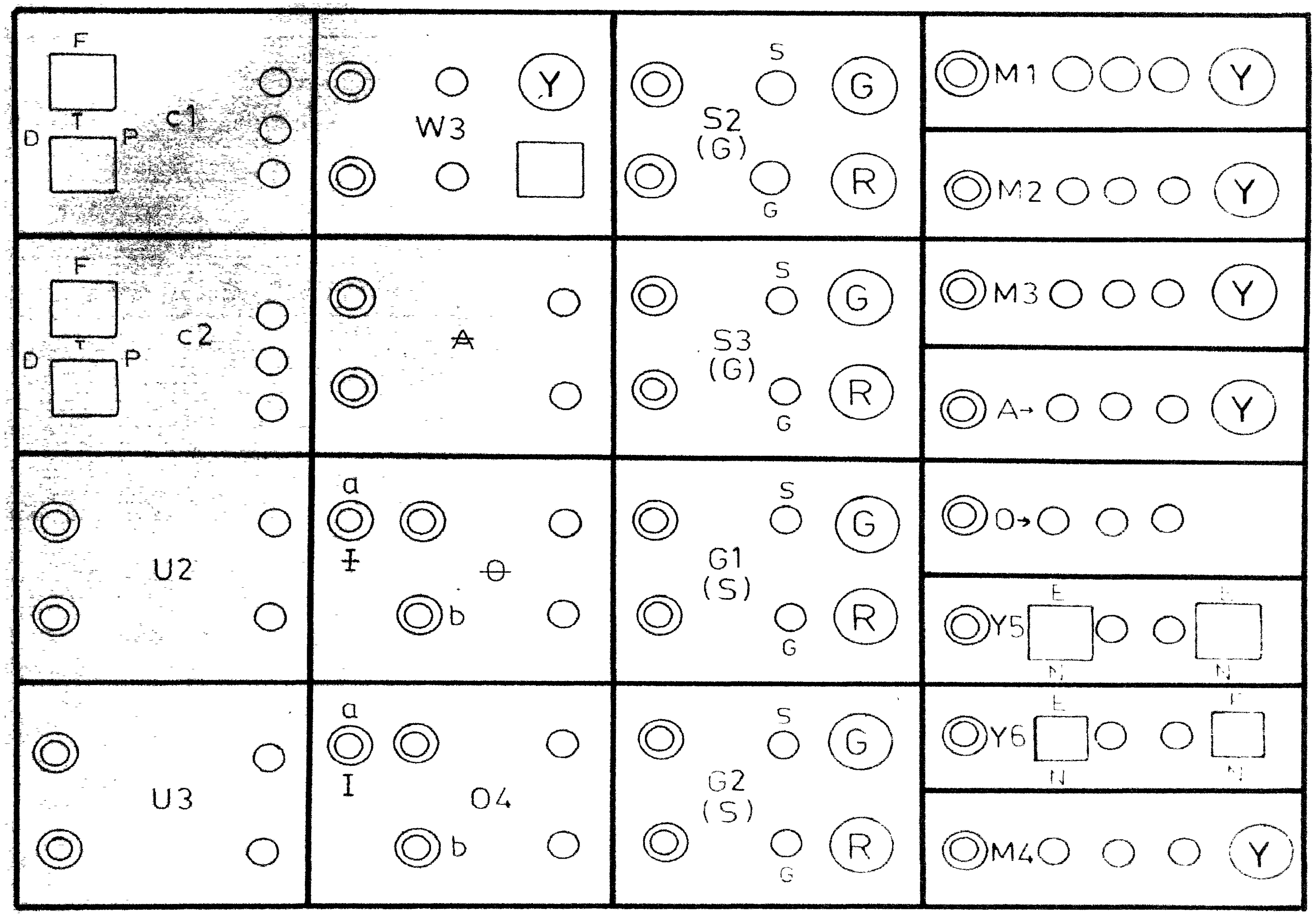
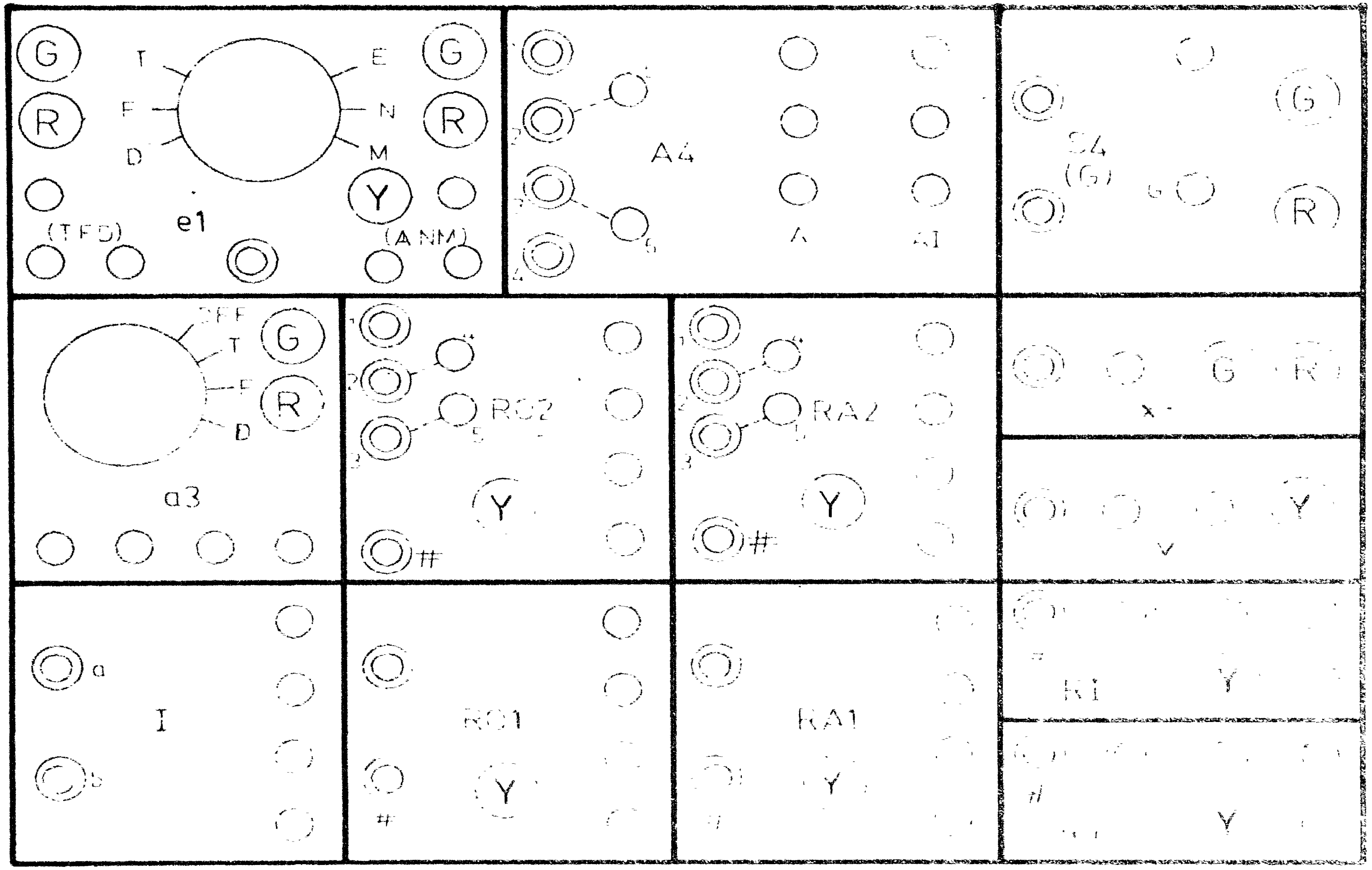


Fig.2



PANEL III

Fig. 3



PANEL IV

all the reverse operators and the sophisticated methods of using the machine for SNS logic all came out as a result of later theoretical studies and partly from trials, in practice, made using ESNY-1. Hence, a more complete instrument with four panels was designed, and this is what is described in this report.

II. Input-Output system

1. Input Module.

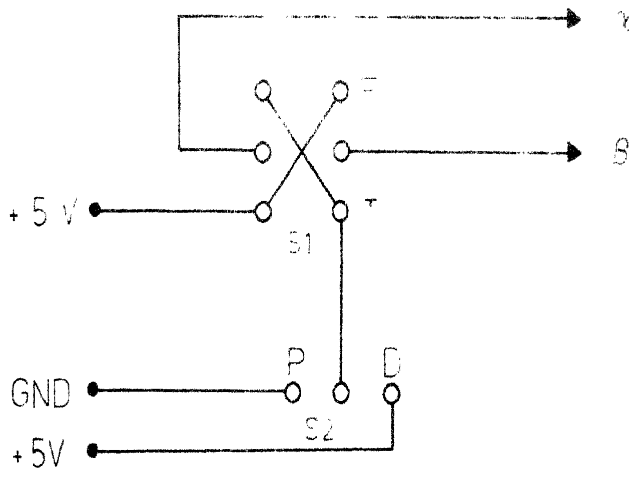
The input module provides inputs for three different logical states: the two lines ($\alpha \quad \beta$) having the values (1 0) corresponding to $\underline{\underline{T}}$, (0 1) corresponding to $\underline{\underline{F}}$, and (1 1) corresponding to $\underline{\underline{D}}$, with 1 corresponding to the existence of a voltage (+5 V in our case) and 0 for absence (gnd, 0 V) for each of the components. The output of this is available in a two-terminal socket and three such are provided, in general, for each input module.

In order to achieve this, two switches, one SPDT and one DPDT, are used in the form indicated in Fig. 5(a). When S2 is set to a P (primitive) state, depending on the setting of S1 the output is T or F. When S2 is set to the D (doubtful) state, the output is D, independent of the setting of S1. For a check the signal of the input module may be connected to the output module (to be described below), which has lamps to indicate the state of the signal. The layout of the input module is as shown for modules a1, a2 in Panel I; b1, b2 in Panel II; c1, c2 in Panel III and a3 in Panel IV, of Figs. 1 to 4. The settings of S1 for T and F and of S2 for P and D are marked in the module a1. (See below for remarks about a3).

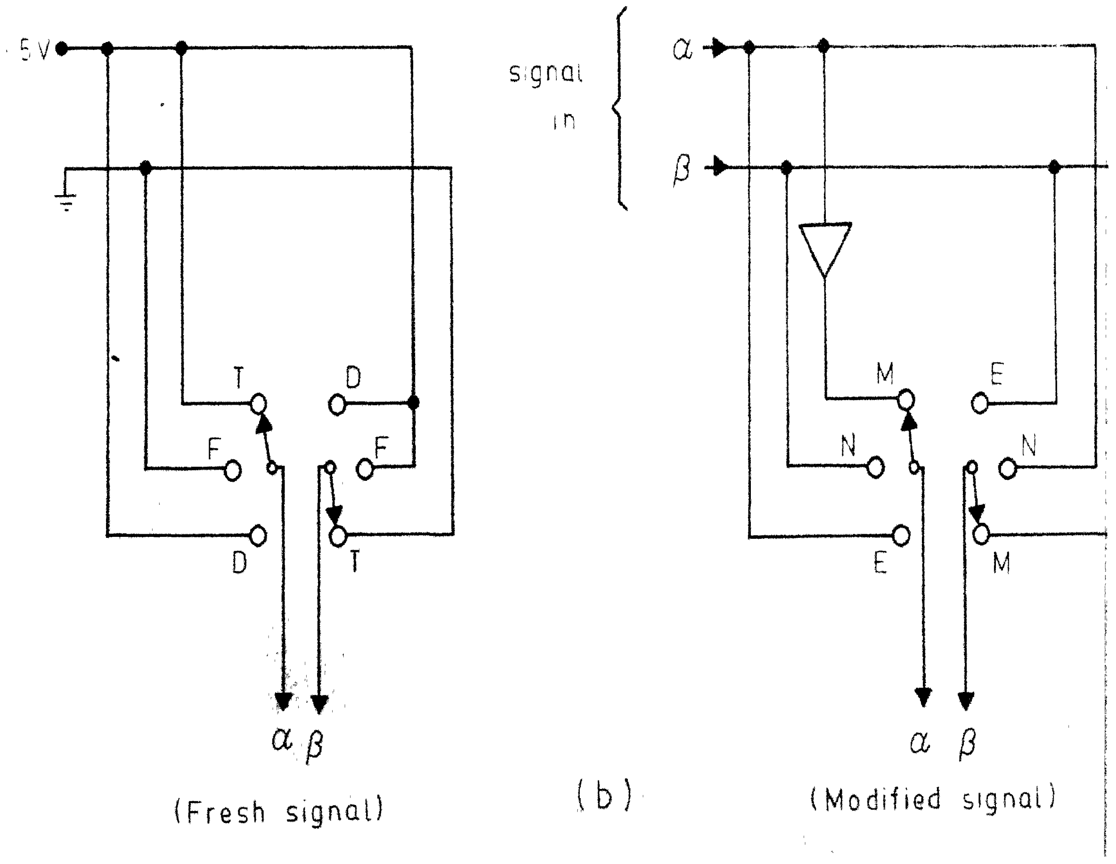
Figure 5 (a) Wiring diagram for setting T and F in S1, and P and D in S2, for the input module.
(b) Wiring diagram using a rotary switch for the input-modifier module.

Since X cannot be set in the input module, we use an additional module (M) (See Sec. III-1) which converts the state (α   β) to (∼α   ∼β) so that a D signal going into M is

..15



(a)



(Fresh signal) (b) (Modified signal)

Fig. 5

output as $\underline{\underline{X}}$. This can be used for obtaining the input signal for the state $\underline{\underline{X}}$, when the need arises.

2. Output module.

The output module is very simple, in that it has one input (two-terminal) socket and three output sockets, and has lamps (red and green) to indicate the state of the signal being fed into it. The layout of the output modules is as shown for $\underline{\underline{x1}}$, $\underline{\underline{x2}}$, in Panel I; $\underline{\underline{y1}}$, $\underline{\underline{y2}}$ in Panel II and $\underline{\underline{x3}}$ in Panel III.

3. Input-modifier module.

Sometimes, there is a necessity for monitoring the input signal, as well as modifying it suitably for testing various consequences for different input states. The method of changing the truth state by means of two switches in the input module is rather complicated for this purpose. Therefore, an intermediate input-modifier module was built, whose layout is as shown for e1 in Panel IV. This can initiate a signal which may be in any one

..17

of the states $\underline{\underline{T}}$, $\underline{\underline{F}}$, $\underline{\underline{D}}$, by a rotary switch setting. In addition, it could also modify an input signal so as to output it <u>via</u> the logical connectives E, N or M given by the Eqs. (1a,b,c):

$$\underline{\underline{a}}\ E = \underline{\underline{a1}}\ ;\quad (\underline{a}_\alpha \longmapsto \underline{a1}_\alpha\ ,\ \underline{a}_\beta \longmapsto \underline{a1}_\beta\ ) \qquad (1a)$$

$$\underline{\underline{a}}\ N = \underline{\underline{a2}}\ ;\quad (\underline{a}_\alpha \longmapsto \underline{a2}_\beta\ ,\ \underline{a}_\beta \longmapsto \underline{a2}_\alpha\ ) \qquad (1b)$$

$$\underline{\underline{a}}\ M = \underline{\underline{a3}}\ ;\quad (\sim\underline{a}_\alpha \longmapsto \underline{a3}_\alpha\ ,\ \sim\underline{a}_\beta \longmapsto \underline{a3}_\beta\ ) \qquad (1c)$$

These different possibilities are set by a multipole rotary switch as shown for e1 in Panel IV. When set at E, the input goes through; when set at N, the negation of it goes through, and when set at M it can serve the purpose of generating the $\underline{\underline{X}}$ state, if necessary, and also can produce the state $\underline{\underline{a}}$ M , which is sometimes needed for various purposes. The circuit diagram of this input-modifier module is given in Fig. 5(b). We may mention, in particular that the operation N interchanges the signals (0 or 1) in the α and β lines, while M involves the introduction of an inverter in each of the lines α and β , so as to implement Eq. (1c) (See Fig. 6).

..18

4. Intermediate output-input modules.

The letters a, b, c with added numerals, as in a1, a2 etc., with letters occuring in the beginning of the alphabet are used for input data. The letters at the end of the alphabet, such as x1, x2, y1, y2 are used for the output, and usually the symbols e1, e2, g1, g2, h1, h2 are used for intermediate terms. The corresponding letters are also used for the modules, as already mentioned. The design of the output module can also be used as a monitor for intermediate steps in an argument when the output of any module can be connected to it and the signal representing its state read off. Such modules are present at g1, g2 of Panel I and h1, h2 of Panel II.

The module a3 in Panel IV has some of the characteristics of the input-output module, in that it has a rotary switch for setting the input at T, F or D (or off), and has also lamps G, R, to indicate the nature of the input.

..19

III. Unary Connectives and their Implementation

As mentioned in [3], the operators which occur in SNS logic are of two types — those that occur as unary connectives e.g. E, N in relations of the type $\underline{\underline{a}}\ E = \underline{\underline{b}}$ or $\underline{\underline{a}}\ N = \underline{\underline{b}}$, and as binary connectives e.g. A, O, in relations of the type $\underline{\underline{a}}\ A\ \underline{\underline{b}} = \underline{\underline{c}}$, $\underline{\underline{a}}\ O\ \underline{\underline{b}} = \underline{\underline{c}}$ where E, N, A, O stand for "equal to", "not", "and", and "or" respectively. We shall not give in this report the definitions of the various connectives, or any theory of their properties, and reference may be made to the publications [2], [4] for such details. However, we shall also use the matrix notation like $\langle \underline{a} | \underline{E} | = \langle \underline{b} |$ and $\langle \underline{a} | \underline{A} | \underline{b} \rangle = \underline{t}(\underline{c})$ occasionally, wherever it is necessary, to make the discussion clear. For these notations also, reference may be made to [4].

1. E, N and M.

The unary operators are those represented by the symbols E, N, M, Y and V, and the logical relations in which they

..20

occur are of the form

$$\underline{\underline{a}}\, Z = \underline{\underline{b}} \quad \text{or} \quad \langle \underline{a} | \underline{Z} | = \langle \underline{b} | \qquad (2)$$

where Z stands for a general operator. The method of implementing these in our analog computer is as follows.

The connective E indicates that the signal in the α and β lines of $\underline{\underline{a}}$ are the same as those of $\underline{\underline{b}}$. It is implemented by merely connecting the two lines of $\underline{\underline{a}}$ to the two lines of $\underline{\underline{b}}$ by a cable, with the two terminals at the input end being connected to the corresponding two terminals at the output end. This is shown schematically in Fig. 6(a).

In the same way, Eq. (1b) for the operator N indicates that the α and β lines of the input $\underline{\underline{a}}$ are interchanged with respect to the corresponding lines of the output $\underline{\underline{b}}$. This again can be implemented very easily by means of a connecting cable, in which the two terminals of the output $\underline{\underline{b}}$ are interchanged with respect to the two terminals of the input $\underline{\underline{a}}$, and this is shown

..21

in Fig. 6(b). We shall call these cable connectors as the E-cable (with the corresponding terms E-connector, E-connection) and N-cable (N-connector, N-connection). The N-cable serves the purpose of obtaining the negated output of any module or to put in a negated input to any module. (The distinction between this type of negation (N) and complementation (M) discussed below is to be noted).

Figure 6. Schematic diagrams of the electrical wiring corresponding to the connectives (a) E, (b) N, (c) M, (d) Y, (e) V.

As regards M, the complementation operation ($\sim 0 = 1$, and $\sim 1 = 0$) is readily implemented by means of inverters. Therefore in the M-module, the input is connected to the output <u>via</u> inverters, as shown in Fig. 6(c). The layout of this module is indicated by M1, M2, M3, M4 in Panel III. Since an input $\underline{\underline{D}}$ into the M-module can produce an output in the $\underline{\underline{X}}$ state, a

..22

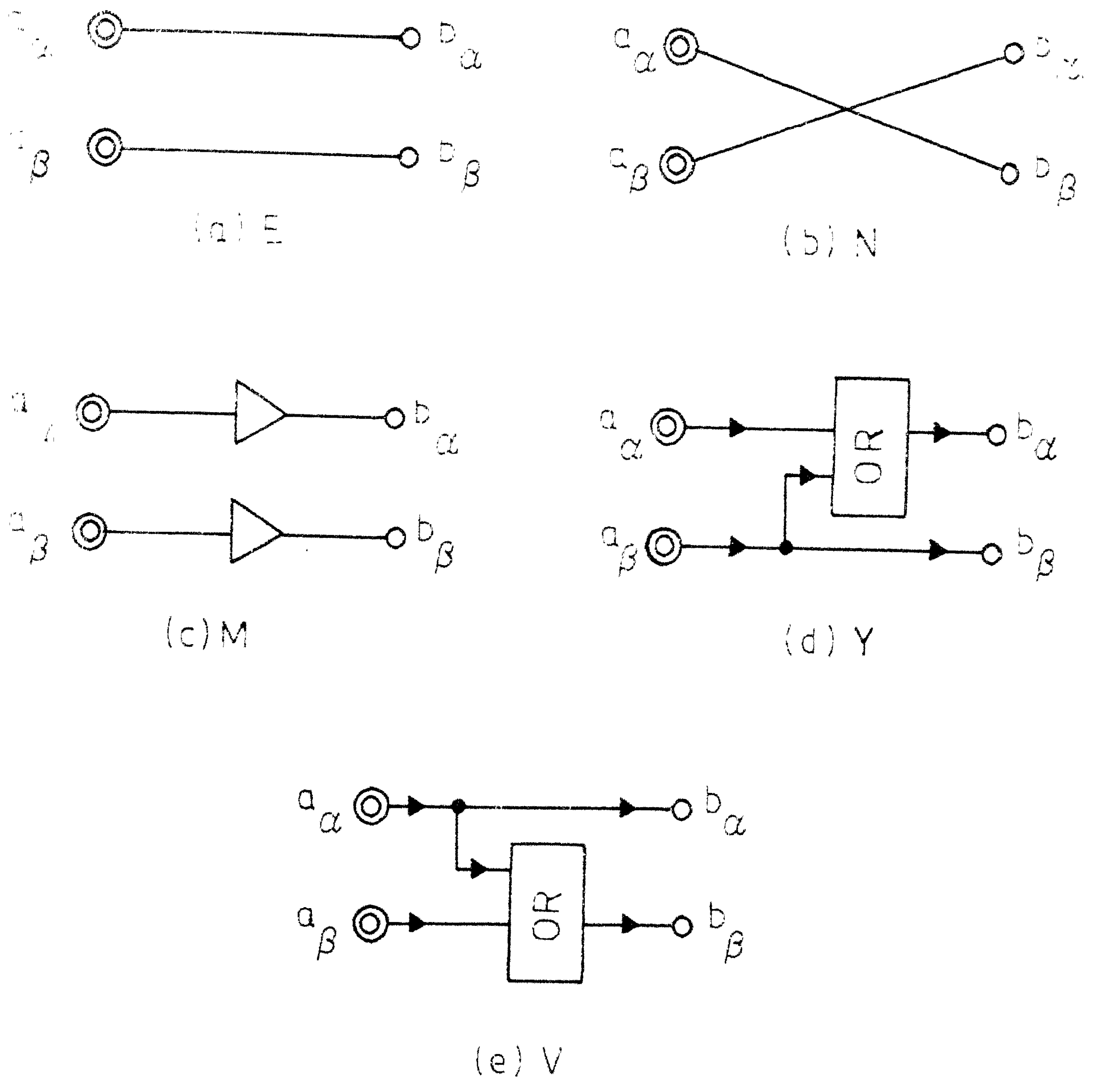


Fig. 6

flashing yellow lamp, with connection to the alarm signal is also provided.

Thus, the connectives N and E , are implemented by means of cable connectors, with and without interchange of the lines at the input and output terminals, and M is implemented via the corresponding M-module.

2. Y and V.

The equation $\underline{\underline{a}}\ Y = \underline{\underline{b}}$, standing for "$\underline{\underline{a}}$ implies $\underline{\underline{b}}$", or "$\underline{\underline{a}} \Longrightarrow \underline{\underline{b}}$", has the consequences $\underline{\underline{a}}_T \longmapsto \underline{\underline{b}}_T$ and $\underline{\underline{a}}_F \longmapsto \underline{\underline{b}}_D$. These can be implemented very simply by having a TTL gate OR, suitably connected as shown in Fig. 6(d). The α-line of $\underline{\underline{a}}$ and the β-line of $\underline{\underline{a}}$ are connected, via an OR, to the α-line of $\underline{\underline{b}}$. The β-line of $\underline{\underline{a}}$ is connected directly to $\underline{\underline{b}}_\beta$. It is readily verified that this circuit has the property required of the connective $\underline{\underline{a}}\ Y = \underline{\underline{b}}$ [4]. The layout of the Y-module is the same as that of M except that no yellow lamp is included, as this module will never give rise to the contradictory state

..24

$\underline{\underline{X}}$ (unless it has $\underline{\underline{X}}$ itself as an input). The modules Y1, Y2 in Panel I, and Y3, Y4 in Panel II have this circuitry.

The connective $\underline{\underline{a}}$ V = $\underline{\underline{b}}$, standing for '$\underline{\underline{a}}$ implicates $\underline{\underline{b}}$', or $\underline{\underline{a}} \Longleftarrow \underline{\underline{b}}$, is required when $\underline{\underline{a}}$ is a necessary condition for $\underline{\underline{b}}$. This has the consequences $\underline{\underline{a}}_F \longmapsto \underline{\underline{b}}_F$ and $\underline{\underline{a}}_T \longmapsto \underline{\underline{b}}_T$. It is possible to show that

$$(\underline{\underline{a}}\ V = \underline{\underline{b}}) \equiv ((\underline{\underline{a}}\ N)\ Y = (\underline{\underline{b}}\ N)) \qquad (3)$$

Hence the connective V is the same as Y, except that the input and output are both negated. Therefore, two of the Y modules (Y5 and Y6 in Panel III), have been modified to have two DPDT switches (S1 and S2) incorporated in them — S1 in the input and S2 in the output, such that each switch can be set either for E or for N. As a consequence, it is possible to get any one of the following four logical connections.

..25

$$[\underline{\underline{a}}\ Y = \underline{\underline{b}}] \equiv [(\underline{\underline{a}}\ N)\ V = (\underline{\underline{b}}\ N)] \tag{4a}$$

$$[(\underline{\underline{a}}\ N)\ Y = \underline{\underline{b}}] \equiv [\underline{\underline{a}}\ V = (\underline{\underline{b}}\ N)] \tag{4b}$$

$$[(\underline{\underline{a}}\ Y)\ N = \underline{\underline{b}}] \equiv [(\underline{\underline{a}}\ N)\ V = \underline{\underline{b}}] \tag{4c}$$

$$[(\underline{\underline{a}}\ N)\ Y)\ N = \underline{\underline{b}}] \equiv [(\underline{\underline{a}}\ N)\ Y = (\underline{\underline{b}}\ N)]$$

$$\equiv [\underline{\underline{a}}\ V = \underline{\underline{b}}] \tag{4d}$$

In the Eqs. (4 a-d), the leftmost equations shows the way in which the settings in the switches are made. For example, in Eq. (4b), S1 is set for N, and S2 is set for E; in Eq. (4c), S1 is set for E and S2 is set for N; and in Eq. (4d), both S1 and S2 are set for N. The last combination, namely with both S1 and S2 set for N, gives rise to the connective V in the equivalent relation $\underline{\underline{a}}\ V = \underline{\underline{b}}$. Hence, to use the Y-module as a V-module, it is only necessary to set the switches S1 and S2 for N. The layout of this Y/V module is shown for Y5, and Y6 in Panel IV.

We can also use the switches in the Y/N module to implement any of the connectives Y, YN, NY, NYN ; V, VN, NV, NVN, as may

..26

be seen from Eqs. (4 a-d). Even without using these switches, we can employ N-connectors suitably for making the connections to the input and output sockets of the Y-module itself, in order to implement any of the equations in (4 a-d).

There are also some other unary connectives in ESNY, namely "unary and" $(A\rightarrow)$, "unary or" $(O\Rightarrow)$, but we are not discussing them now, but shall only consider them while discussing, in general, binary reverse operators, to which these are closely related.

It is a very interesting observation that, although the design has been specifically made only to take care of the primary input states $\underline{\underline{T}}$ and $\underline{\underline{F}}$ for all the above-mentioned unary connectives, (the output being $\underline{\underline{T}}$, $\underline{\underline{F}}$ or $\underline{\underline{D}}$), all the circuits work equally well with $\underline{\underline{D}}$ inputs, and the corresponding output, as given by the circuit, agrees with what is expected from logical theory, or from the vector-matrix formalism developed by us [4]. Briefly, the reason for this arises from the fact that the

..27

matrix algebra employed is a linear Boolean algebra, with $\underline{\underline{T}}(1 \quad 0)$ and $\underline{\underline{F}}\ (0 \quad 1)$ as the basic states. Consequently, since the $\underline{\underline{D}}$ state is only a sum ($\oplus$) of the states $\underline{\underline{T}}$ and $\underline{\underline{F}}$ i.e., $\underline{\underline{T}} \oplus \underline{\underline{F}} = \underline{\underline{D}}$, the circuit also adds up the outputs corresponding to the two elements $(\alpha \quad \beta)$ of the inputs by the Boolean addition formula, which makes it automatically give correct results for $\underline{\underline{D}}$ inputs also.

However, a corresponding result does not occur for $\underline{\underline{X}}$, which can be defined as $\underline{\underline{T}} \otimes \underline{\underline{F}} = \underline{\underline{X}}$, and an $\underline{\underline{X}}$ input need not necessarily lead to the expected consequence for the logic of the $\underline{\underline{X}}$ input from the circuits of the type designed by us to work for the $\underline{\underline{T}}$ and $\underline{\underline{F}}$ states. The way in which the $\underline{\underline{X}}$ input is taken care of is discussed in Sec. IV-2, dealing with the binary connectives A, O, W and U, where we explain the use of the so-called "X-priority rule", which requires that, whenever the state $\underline{\underline{X}}$ is an input, the output will also necessarily be

..28

$\underline{\underline{X}}$ (because, a logical impossibility fed in will again lead only to an impossibility, and it cannot normally lead to a useful or "allowed" value of the truth state such as $\underline{\underline{T}}$, $\underline{\underline{F}}$, or $\underline{\underline{D}}$). This rule, however, has exceptions (See Sec.IV-1).

IV. Binary Forward Connectives and their Implementation.

1. Binary "and" and "or".

The one letter symbol for "and" is A and for "or" is O. These are the most important among the classical binary connectives. It is well known that these operations have a close analogy to the Boolean operations of multiplication and addition for a single Boolean variable, having the states 0 and 1 corresponding to $\underline{\underline{F}}$ and $\underline{\underline{T}}$ in logic. However, the signal for a logical state is kept in two lines, α and β, in our SNS logic machine. Therefore, we must find out the laws giving the connections between the α and β components of $\underline{\underline{a}}$ and $\underline{\underline{b}}$, and the corresponding components $\underline{\underline{c}}_\alpha$ and $\underline{\underline{c}}_\beta$, for a binary relation $\underline{\underline{a}}\ Z\ \underline{\underline{b}} = \underline{\underline{c}}$.

..29

This is discussed in detail in [4], and we state here the main results for the two connectives A and O. This may be seen readily in the following definitions for A and O, in Eqs. (5a) and (5b), in which we shall represent the classical one-element logical connectives by the symbols AND, OR and NOT (using capital letters). We shall also refer to these as classical logic (CL) connectives. Thus,

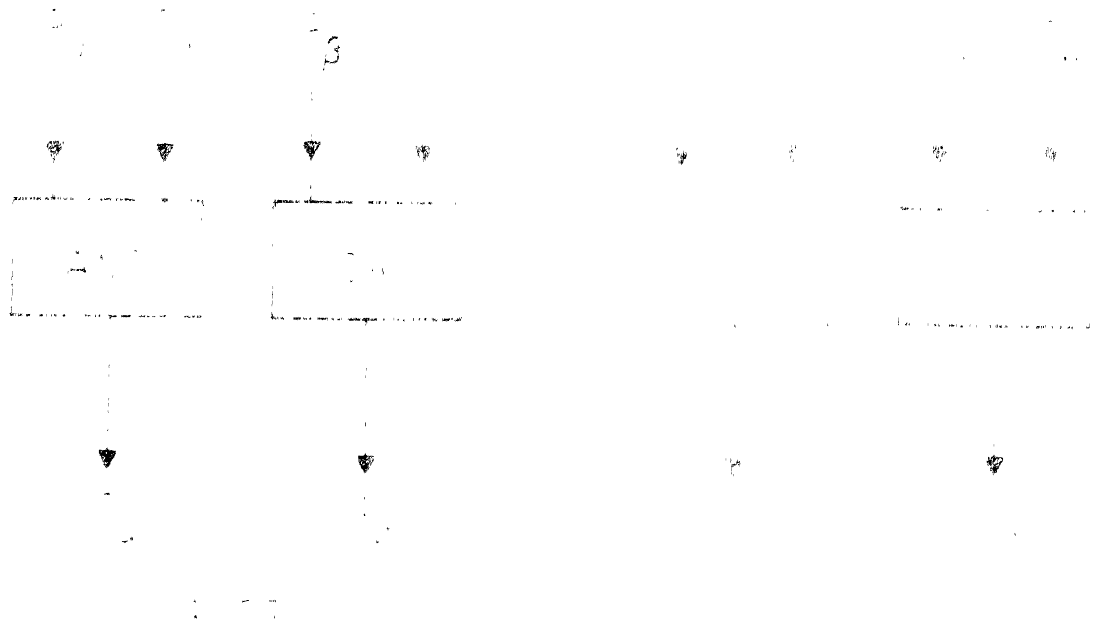
$$(\underline{\underline{a}}\ \mathrm{A}\ \underline{\underline{b}} = \underline{\underline{c}}) \equiv (\underline{a}_\alpha \text{ AND } \underline{b}_\alpha = \underline{c}_\alpha\ ,\ \underline{a}_\beta \text{ OR } \underline{b}_\beta = \underline{c}_\beta\ ) \tag{5a}$$

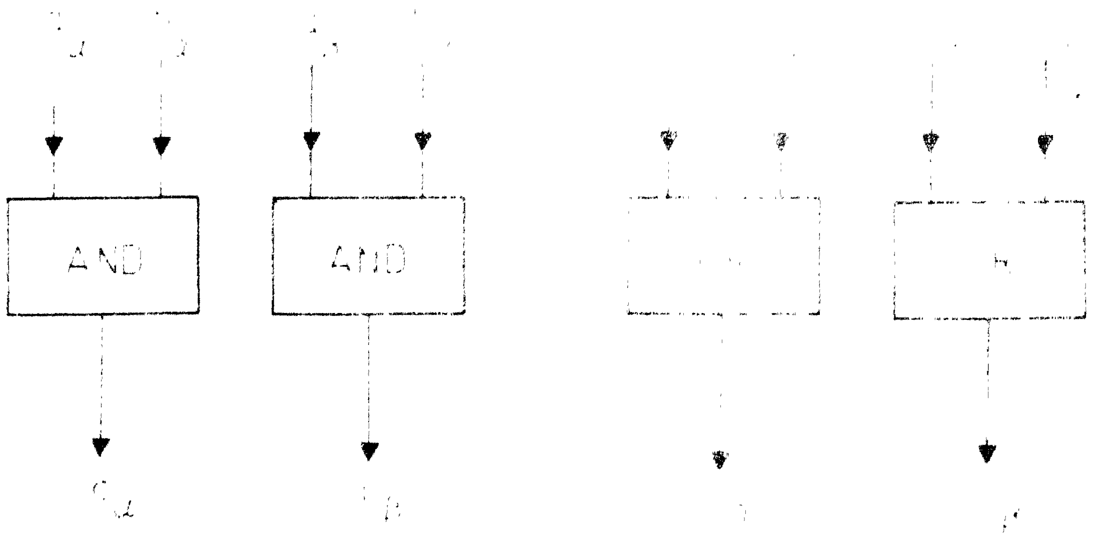
$$(\underline{\underline{a}}\ \mathrm{O}\ \underline{\underline{b}} = \underline{\underline{c}}) \equiv (\underline{a}_\alpha \text{ OR } \underline{b}_\alpha = \underline{c}_\alpha\ ,\ \underline{a}_\beta \text{ AND } \underline{b}_\beta = \underline{c}_\beta\ ) \tag{5b}$$

The way in which these are implemented in the electrical connections is by the use of AND and OR gates (which themselves could be wired by using only a NAND gate and inverters). Fig. 7 (a) and (b) show the principle of the wiring connections of

Figure 7. Schematic diagrams for the SNS connectives A, O, W, U giving the outputs $\underline{c}_\alpha$ and $\underline{c}_\beta$ in terms of the inputs $\underline{a}_\alpha$, $\underline{a}_\beta$, $\underline{b}_\alpha$, $\underline{b}_\beta$.

..30





(c) with

Fig

the SNS connectives "and" and "or", in terms of classical connectives AND and OR. The figures in 10(a) and 10(b) (Sec. IV-2) show how these are implemented, using the readily available TTL NAND gates and inverters. The description of these circuits in terms of CL connectives is given in Eqs. (6a) and (6b), and we shall use this style for describing all other SNS connectives.

$$\underline{\underline{a}}\ \mathrm{A}\ \underline{\underline{b}} = \underline{\underline{c}}$$

$$\text{NOT } (\underline{\underline{a}}_\alpha \text{ NAND } \underline{\underline{b}}_\alpha) = \underline{\underline{c}}_\alpha \quad ; \quad (\text{NOT } \underline{\underline{a}}_\beta) \text{ NAND } (\text{NOT } \underline{\underline{b}}_\beta) = \underline{\underline{c}}_\beta \qquad (6a)$$

$$\underline{\underline{a}}\ \mathrm{0}\ \underline{\underline{b}} = \underline{\underline{c}}$$

$$(\text{NOT } \underline{\underline{a}}_\alpha) \text{ NAND } (\text{NOT } \underline{\underline{b}}_\alpha) = \underline{\underline{c}}_\alpha \quad ; \quad \text{NOT } (\underline{\underline{a}}_\beta \text{ NAND } \underline{\underline{b}}_\beta) = \underline{\underline{c}}_\beta \qquad (6b)$$

The layout of the "and" and "or" modules are very simple as shown in A1 and 01 of Panel I. The two input sockets ($\underline{\underline{a}}$ and $\underline{\underline{b}}$) are shown by double circles, and the output is available from either of the two sockets on the right hand side. The "binary and" (A) modules are available as A1, A2, and "binary or" (0) modules are available as 01, 02 in Panel I.

..32

We have already mentioned that, whenever an $\underline{\underline{X}}$ is the input $\underline{\underline{a}}$ or $\underline{\underline{b}}$ in a binary logical relation of the form:

$$\underline{\underline{a}}\ Z\ \underline{\underline{b}} = \underline{\underline{c}}$$

where Z is a general connective operator, the output is once again X itself except for SNS operators. Unfortunately, many of the circuits described below, (and also the A and O, described above), do not have this property, although they give the correct output for $\underline{\underline{D}}$ inputsof $\underline{\underline{a}}$ and $\underline{\underline{b}}$, although the circuit has been designed essentially for the inputs $\underline{\underline{T}}$ and $\underline{\underline{F}}$ only. Hence, a separate circuit called "$\underline{\underline{X}}$ - check circuit" has been designed and incorporated. What this effectively does is shown in Fig.8. In this, the output from the connective Z1 (or an input $\underline{\underline{a}}$, as the case may be), goes in as the $\underline{\underline{a}}$-input for the connective Z. Similarly, either as an output of the connective Z2, or as a fresh input $\underline{\underline{b}}$, the signal comes in as the $\underline{\underline{b}}$-input to Z. Normally, the output of Z, marked $\underline{\underline{c}}$', would be the output that is taken,

..33

and it will be correct for the states $\underline{\underline{T}}$, $\underline{\underline{F}}$, $\underline{\underline{D}}$ of $\underline{\underline{a}}$ and $\underline{\underline{b}}$. If, however, one of $\underline{\underline{a}}$ or $\underline{\underline{b}}$ is $\underline{\underline{X}}$, then we make use of the $\underline{\underline{X}}$-check module (XC) shown in the bottom right hand side of Fig. 8. This has, as inputs, the signals $\underline{\underline{a}}$, $\underline{\underline{b}}$ and $\underline{\underline{c}}'$. It then checks for the occurence of $\underline{\underline{X}}$ either in $\underline{\underline{a}}$ or in $\underline{\underline{b}}$, and if so, gives the state $\underline{\underline{X}}$ in its output socket marked $\underline{\underline{c}}$. Otherwise, it allows the output $\underline{\underline{c}}'$ of Z to go through unchanged to the output socket $\underline{\underline{c}}$.

The bottom right hand side of Fig. 8 gives also the layout of the XC-module. The circuit of the XC-module is shown in Fig. 9(a) and (b). Fig. 9(a) contains the circuit which checks both $\underline{\underline{a}}$ and $\underline{\underline{b}}$ separately for the signal (0 0). If either is so, then the NOR will give an output 0; otherwise the output is 1. This signal (δ) is then fed to the two AND's in the circuit shown in Fig. 9(b), whose other inputs comprise of $\underline{c}'_\alpha$ and $\underline{c}'_\beta$ respectively. It is readily verified that if δ is non-zero, then the output ($\underline{c}_\alpha$ $\underline{c}_\beta$) will be the same as ($\underline{c}'_\alpha$ $\underline{c}'_\beta$), while if δ is zero, the output $\underline{\underline{c}}$ is (0 0), or an $\underline{\underline{X}}$ state.

..34

This circuit is provided as an extra facility to ESNY, so that it may be used in combination with any module which has the circuit of a binary connective Z. By a detailed study, it has been found that some of the binary connectives, like W, automatically have the **X**-check property. For others, the output **c**' of Z, is not directly taken, but is put through the **X**-check module, and the checked signal **c** is taken for further utilization.

Figure 8. Schematic diagram showing how a signal **a** Z **b** = **c** is modified by the **X**-check module to give the **X** checked output **c**.

Figure 9. (a) Circuit for obtaining the **X**-check signal (δ)..

(b) The **X**-check circuit to obtain **c** from **c**'.

..35

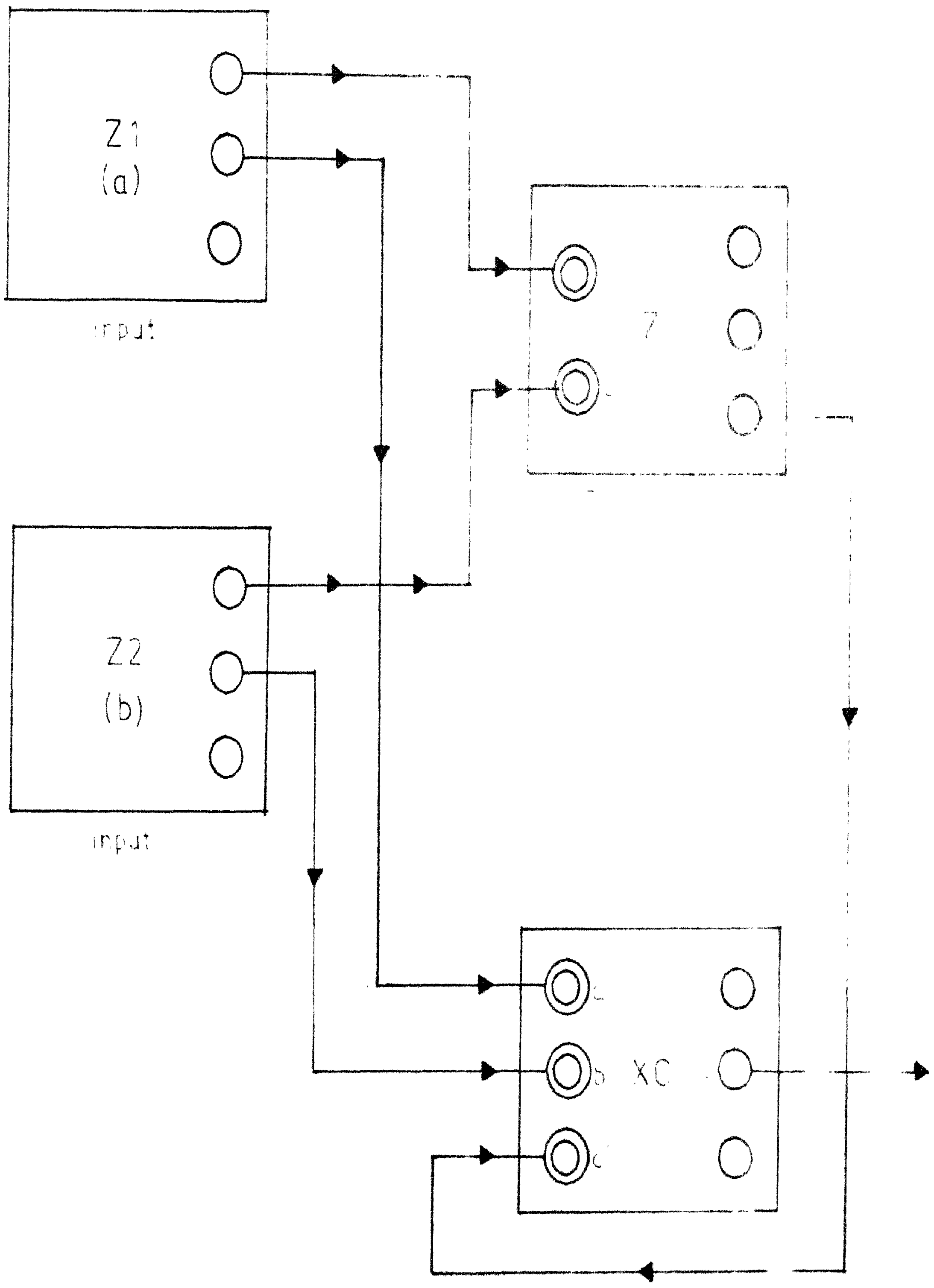
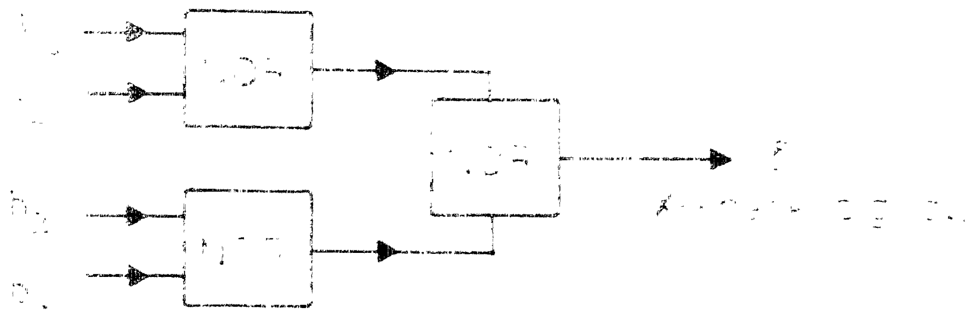
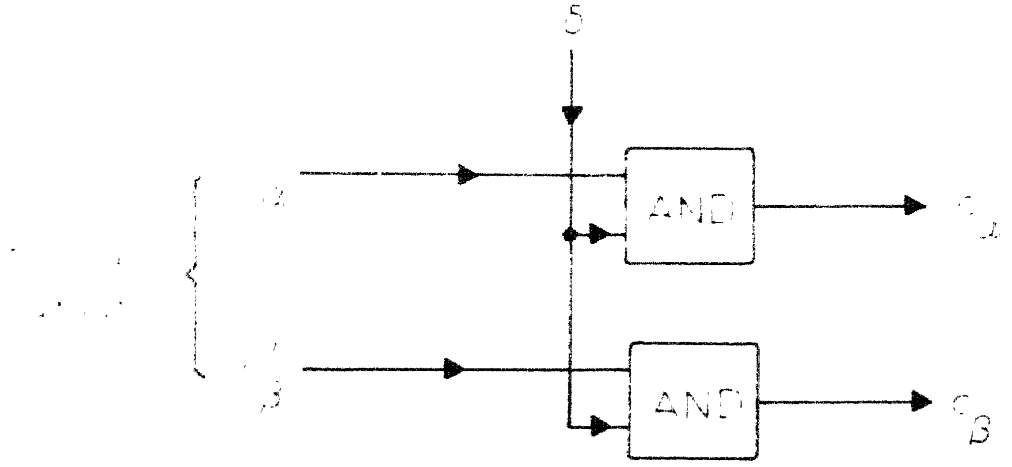


Fig. 8





(b)

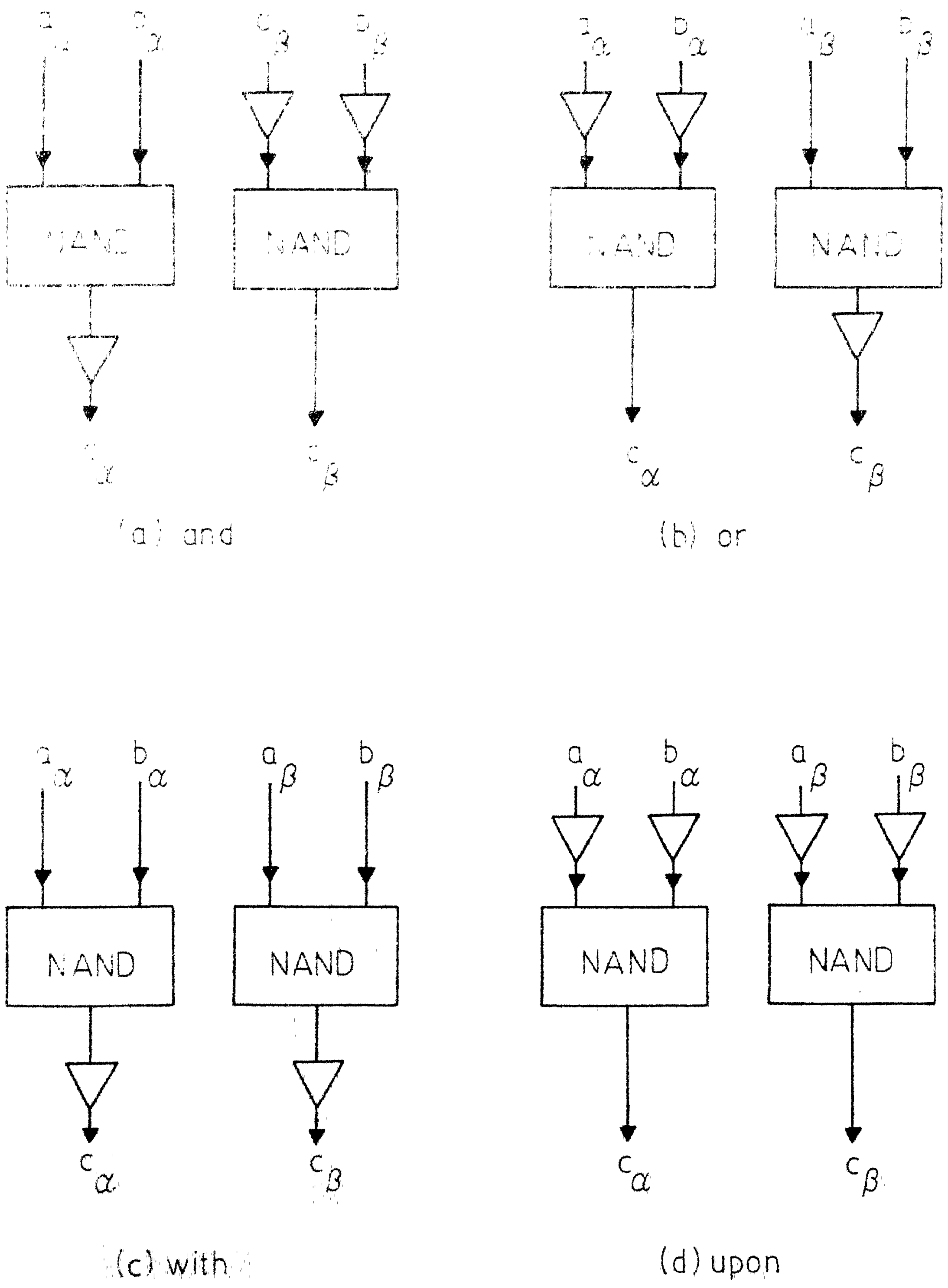
Fig 9

2. Binary connectives "with" and "upon".

Figure 10. Diagrams giving more details of the circuit diagrams given in Fig. 7, using only NAND gates and inverters. Note that AND and OR can both be implemented by means of these elements.

The two schematic diagrams in Figs. 7(a) and (b), indicate that we have used the combinations (AND, OR) and (OR, AND) for the α and β lines of the connectives A and O respectively. This suggests that two other similar combinations could be thought of, namely (AND, AND) and (OR, OR). These are shown in Figs. 7(c) and (d), and their possible detailed implementation <u>via</u> NAND gates is indicated in Figs. 10(c) and (d). The newly defined connectives, corresponding to these circuits, have been named "with" (W) and "upon" (U). They have interesting properties, and they can be defined only in SNS logic and not in classical logic. Although, from the point of view of

..38



a_α
b_α
a_β
b_β
NAND
c_α
c_β
(a) and
(b) or
(c) with
(d) upon

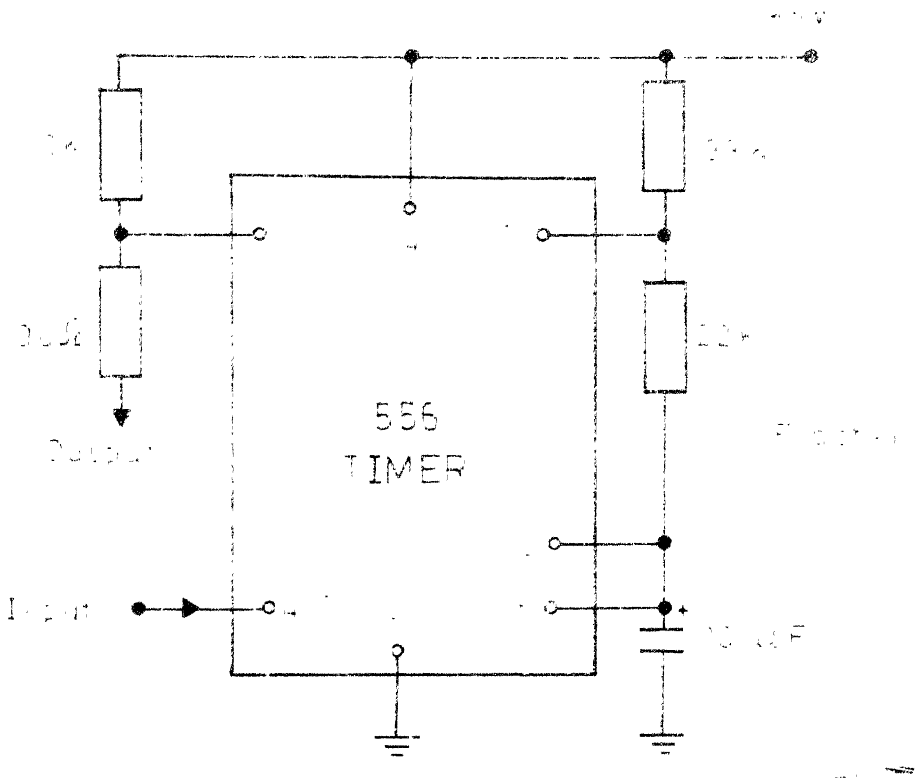
Fig 10

their circuits, W and U are closely related to A and O, their properties are very different. (See [3] and [4]). In particular, W checks for the consistency of two states a and b and gives the same state as output if the two agree, i.e. if they are both T, both F or both D. On the other hand, if one is T and the other is F, it will output an X for inconsistency, flash a yellow lamp and sound an alarm. The circuits for these are shown in Fig.11 (a) and (b) respectively. In this way, if

Figure 11. Diagrams of (a) the flasher, and (b) the alarm circuits, using the 556 timer chip.

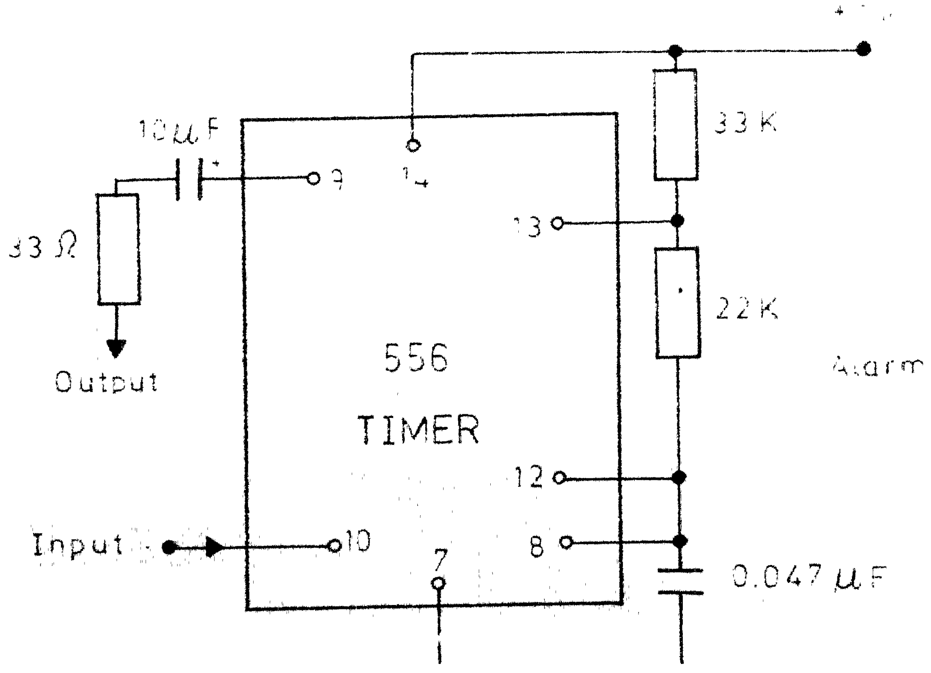
at any time, it is necessary to check whether two statements have the same state or opposite states, this module can be used. However, if one input is in a definite "primitive" state (T or F), while the other is in the D state, only the primitive state will come through the module W, quite unlike what happens with A and O. The use of W will be illustrated in Sec. VIII,

..39



556
TIMER
Input
Output
33 Ω
10 μF

(a)



10 μF
9
14
33 Ω
Output
13
33 K
22 K
556
TIMER
Alarm
12
Input
10
7
8
0.047 μF

Fig.11

while dealing with applications.

Similarly, U is a module used for checking for unanimity In this case also, if both inputs agree, the same will be the output; but if they disagree in any way, only $\underline{\underline{D}}$ will come out. The behaviour of W and U have not been designed arbitrarily, but follow straightaway from the definitions given below in Eqs. (7) and (8) analogous to Eqs. (5a) and (5b) for A and O:

$$(\underline{\underline{a}}\ W\ \underline{\underline{b}} = \underline{\underline{c}}) \longmapsto (\underline{a}_\alpha \text{ AND } \underline{b}_\alpha = \underline{c}_\alpha\ ,\ \underline{a}_\beta \text{ AND } \underline{b}_\beta = \underline{c}_\beta) \quad (7)$$

$$(\underline{\underline{a}}\ U\ \underline{\underline{b}} = \underline{\underline{c}}) \longmapsto (\underline{a}_\alpha \text{ OR } \underline{b}_\alpha = \underline{c}_\alpha\ ,\ \underline{a}_\beta \text{ OR } \underline{b}_\beta = \underline{c}_\beta) \quad (8)$$

The layout of the W module is shown in W1 in Panel I. It is similar to that of the A-module, except that a yellow lamp is included for indicating inconsistencies. The U-module, as in U1, is identical in its layout to the A and O modules, since it has no need for a yellow lamp.

3. Binary S and G.

The name S is used to stand for the word "same" or "similar"

..41

It is defined, such that two states $\underline{\underline{a}}$ and $\underline{\underline{b}}$ connected by the relation ($\underline{\underline{a}}$ S $\underline{\underline{b}}$ = $\underline{\underline{c}}$) will have a truth value $\underline{\underline{T}}$ if $\underline{\underline{a}}$ and $\underline{\underline{b}}$ have the same state, and a truth value $\underline{\underline{F}}$ if they have opposite states. It should be noted that this is true only for the classical states $\underline{\underline{T}}$ and $\underline{\underline{F}}$ used as inputs for $\underline{\underline{a}}$ and $\underline{\underline{b}}$. In SNS logic, when $\underline{\underline{D}}$ state inputs can also occur, the connective S can give the state $\underline{\underline{D}}$, also for the output $\underline{\underline{c}}$, if either $\underline{\underline{a}}$, or $\underline{\underline{b}}$, or both are $\underline{\underline{D}}$. These properties can be readily implemented by the circuit shown in Fig. 12.

Figure 12. Schematic diagram of the implementation of the connective "similar", in the S-module.

In this circuit, the set of two XOR's (i) and (ii) check that the α-components of $\underline{\underline{a}}$ and $\underline{\underline{b}}$ are the same, and similarly for the β-components of $\underline{\underline{a}}$ and $\underline{\underline{b}}$. If the output of both of these are 1, the AND (iii) will give a signal 1. In effect, this circuit checks for the identity of the SNS states of the

..42

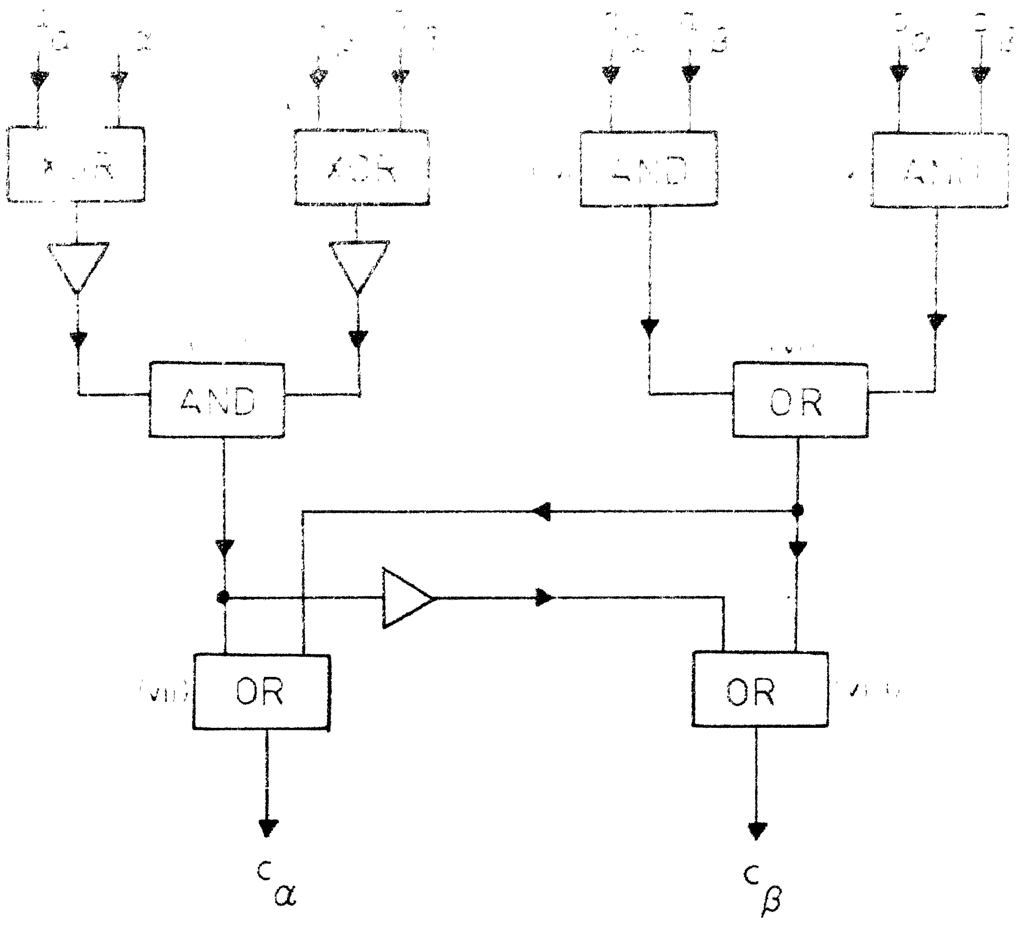


Fig.12."Similar"

two inputs $\underline{\underline{a}}$ ($\underline{\underline{a}}_\alpha$ $\underline{\underline{a}}_\beta$) and $\underline{\underline{b}}$ ($\underline{\underline{b}}_\alpha$ $\underline{\underline{b}}_\beta$). Such a check is itself a very useful logical test, and we have given this operator the name "agree" (G). This circuit is shown separately in Fig.13, where $\underline{\underline{c}}_\alpha$ is the output mentioned above, and $\underline{\underline{c}}_\beta$ is obtained by inverting this signal.

Figure 13. Schematic diagram of the "agreement" operator G.

Hence, for the relation

$$\underline{\underline{a}}\ G\ \underline{\underline{b}} = \underline{\underline{c}}$$

$\underline{\underline{c}}$ will be $\underline{\underline{T}}$ if $\underline{\underline{a}}$ and $\underline{\underline{b}}$ are the same among the four possible states $\underline{\underline{T}}$, $\underline{\underline{F}}$, $\underline{\underline{D}}$, $\underline{\underline{X}}$. If the two are different, then $\underline{\underline{c}} = \underline{\underline{F}}$.

Thus the circuit for G forms the left-hand part of the circuit for S. The right-hand part of the S-circuit, only serves the purpose of testing whether either $\underline{\underline{a}}$, or $\underline{\underline{b}}$, or both, is a $\underline{\underline{D}}$ and giving a signal 1, when this occurs. This is done by the two AND's (iv) and (v) which will give a signal 1 when

..44

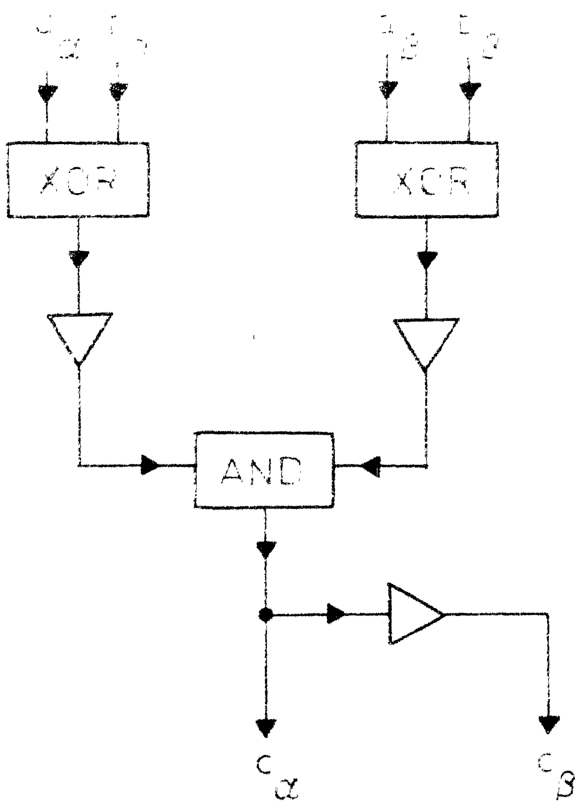


Fig.13."Agree"

$\underline{\underline{a}}$ and $\underline{\underline{b}}$ are $\underline{\underline{D}}$. The combination of these by the OR in (vi) gives the required result. Now the outputs from the left half and from the right half of Fig. 12 are suitably combined by the two OR's (vii) and (viii), so that ($\underline{c}_{\alpha}$ $\underline{c}_{\iota}$) has all the properties required of the connective S (whose full truth table is available in Sec. VIII-1 (Table 4b).

The layout of the modules for G and S follow the same pattern as for the other binary connectives discussed earlier, like A and O. However, in order that we may check the same two inputs either for similarity (according to S) or for agreement (according to G), several G(S) and S(G) modules have been provided. Thus, S, in Panel II, checks for similarity and gives a green light if it is $\underline{\underline{T}}$. In Panel III, there are two S(G)'s and two G(S)'s. The inputs on the left are the same in both types. The output on the right can be taken from the socket marked S, or G, as the case may be, in all of them. However, the lamps will show only the signal corresponding to the unbrack$\overset{\text{te}}{\wedge}$ed symbol — viz., S for S2(G), and G for G1(S).

4. Binary implication I.

The binary connective I has the property that it is $\underline{\underline{F}}$ only if $\underline{\underline{a}}$ is $\underline{\underline{T}}$ and $\underline{\underline{b}}$ is $\underline{\underline{F}}$, and is $\underline{\underline{T}}$ otherwise. In this sense, it is very similar to an "or", and can be represented by the equation

$$\underline{\underline{a}}\ \mathrm{I}\ \underline{\underline{b}} = (\underline{\underline{a}}\ \mathrm{N})\ \mathrm{O}\ \underline{\underline{b}} \qquad (9)$$

where $\underline{\underline{a}}$N stands for "$\underline{\underline{a}}$-negated". Eq. (9) is known very well in logic and this is precisely what is employed in the circuit shown in Fig.14. An example of the module is seen in I1 of Panel IV. Unlike A, O, W, U, G, S, in which the inputs $\underline{\underline{a}}$ and $\underline{\underline{b}}$ are interchangeable, it is absolutely necessary to state which is the $\underline{\underline{a}}$-input and which is the $\underline{\underline{b}}$-input, for I. Hence the two inputs are differentiated as $\underline{\underline{a}}$ and $\underline{\underline{b}}$ in the I-modules.

Since O and I are closely related, the module O(I)4 of Panel III, is wired so that it can be used either as an O, or

..47

as an I, by connecting the inputs to the appropriate sockets on the module.

Figure 14. Schematic diagram for the connective I in the equation $\underline{\underline{a}}$ I $\underline{\underline{b}}$ = $\underline{\underline{c}}$. Note its close similarity to that of O.

V. Binary Reverse-Operators and Related Connectives.

1. Reverse "and" and reverse "or".

The definition of a reverse operator may be illustrated by the example of "reverse and" ($\overleftarrow{A}$). Here the following two equations are equivalent :

$$(\underline{\underline{a}}\ A\ \underline{\underline{b}} = \underline{\underline{c}}) \equiv (\underline{\underline{c}}\ \overleftarrow{A}\ \underline{\underline{a}} = \underline{\underline{b}}) \qquad (10)$$

When the reverse operator is applied, it is very important to distinguish between the first input which is $\underline{\underline{c}}$, and the second input which is $\underline{\underline{a}}$, in order to give the outcome $\underline{\underline{b}}$. The reverse operator is definable only in terms of the corresponding forward operator (whose equation is given on the left-hand side of Eq. 10)

..48

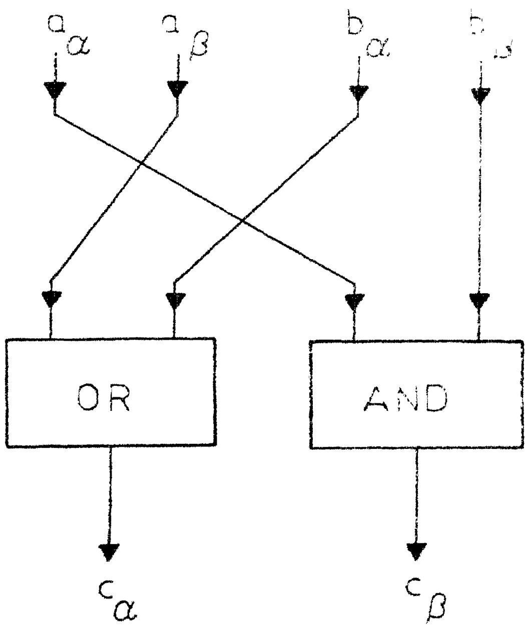


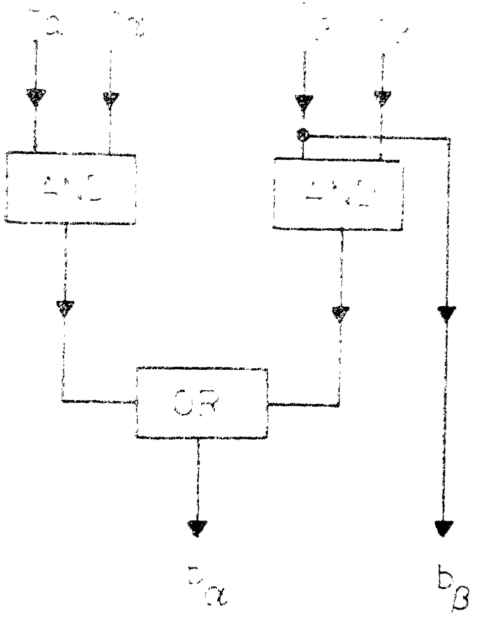
Fig 14. Binary imply

We can define reverse operators corresponding to four common forward operators A, O, I, J. The truth tables for all of these are given in Table 1. These have been implemented in electronic circuits using TTL gates, and the schematic diagrams of all of these are shown in Fig. 15.

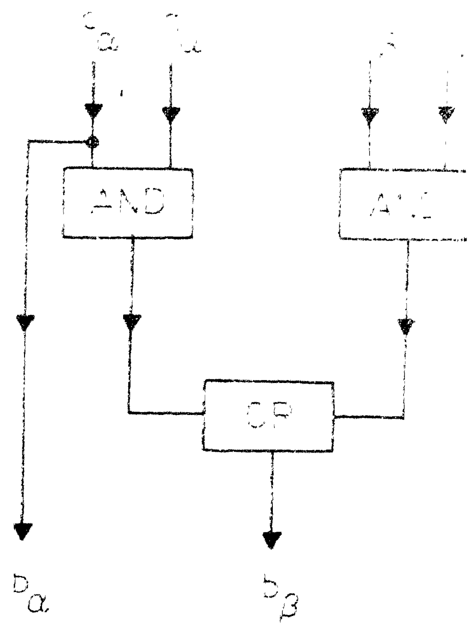
Figure 15. Schematic diagrams of the reverse operators (a) $\overleftarrow{A}$, (b) $\overleftarrow{O}$, (c) $\overleftarrow{I}$, (d) $\overleftarrow{J}$

The circuit for reverse "and" ($\overleftarrow{A}$) and reverse "or" ($\overleftarrow{O}$) are somewhat similar, in that both of them have $\underline{c}_\alpha$, $\underline{a}_\alpha$ and $\underline{c}_\beta$, $\underline{a}_\beta$ as inputs to AND gates and the outputs of these are combined _via_ an OR. The only difference is the way in which $\underline{b}_\alpha$ and $\underline{b}_\beta$ are connected. It is readily verified that these circuits provide the correct outputs for the input states $\underline{\underline{T}}$, $\underline{\underline{F}}$ and $\underline{\underline{D}}$ for $\underline{\underline{c}}$, $\underline{\underline{a}}$ as given in Table 1 (they do not give the correct output when there is an $\underline{\underline{X}}$ input into the circuit, and the $\underline{\underline{X}}$-check module will have to be used for this purpose).

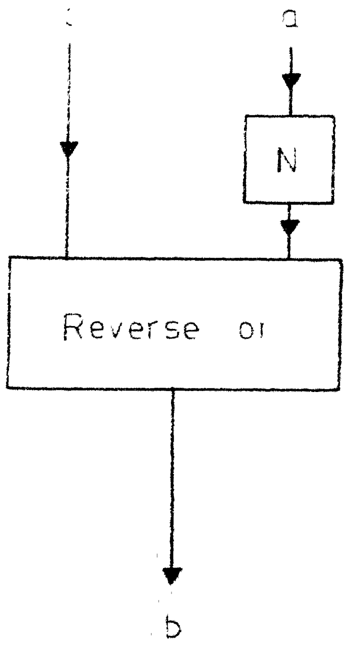
..50



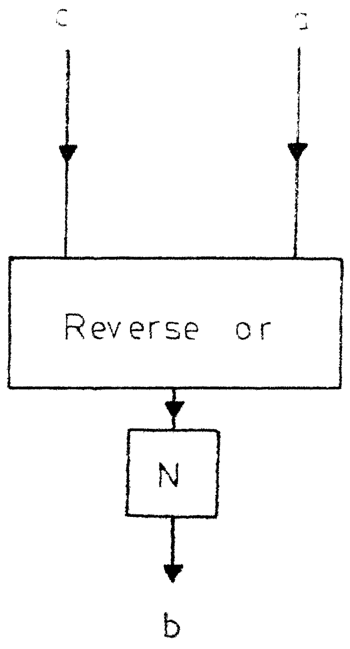
(a) Reverse and



(b) Reverse or



(c)Reverse imply



(d)Reverse implicate

Fig.15

Table 1. Truth table for the reverse connectives $\overleftarrow{A}$, $\overleftarrow{O}$, $\overleftarrow{I}$ and $\overleftarrow{J}$.

2. Reverse "implies" and reverse "implicates".

In the case of I and J the reverse operator is best defined in terms of $\overleftarrow{O}$, since the forward relations involving I and J can be obtained using only O and negating the input or the output. The relevant equation for I is

$$\underline{\underline{a}}\ I\ \underline{\underline{b}} = (\underline{\underline{a}}\ N)\ O\ \underline{\underline{b}} = \underline{\underline{c}} \qquad (11a)$$

which immediately leads to the reverse relation

$$\underline{\underline{c}}\ \overleftarrow{I}\ \underline{\underline{a}} = \underline{\underline{c}}\ \overleftarrow{O}\ (\underline{\underline{a}}\ N) \qquad (11b)$$

This suggests the schematic diagram of the circuit as shown in Fig. 15(c). Here the circuit for reverse "or" is shown as a block, and only the addition of the negation N to the input $\underline{\underline{a}}$ is marked.

In the same way, for "implicates" (J), we have the equation

..52

Table 1

a	b	a Z c = b for			
		A	O	I	J
T	T	T	D	T	D
T	F	X	I	D	F
F	T	F	X	F	X
F	F	D	F	X	T
T	D	T	D	D	D
D	T	D	D	D	D
F	D	D	F	F	T
D	F	D	D	D	D
D	D	D	D	D	D

Table 2

a	a A = b	a O = b
T	T	D
F	X	T
D	T	D

$$\underline{\underline{a}}\ J\ \underline{\underline{b}} = \underline{\underline{a}}\ O\ (\underline{\underline{b}}\ N) = \underline{\underline{c}} \qquad (12a)$$

which gives for the reverse operator $\overleftarrow{J}$,

$$\underline{\underline{c}}\ \overleftarrow{J}\ \underline{\underline{a}} = (\underline{\underline{c}}\ \overleftarrow{O}\ \underline{\underline{a}})\ N \qquad (12b)$$

The corresponding schematic circuit is shown in Fig.15(d), and in this case it is only necessary to add a negation N in the output $\underline{\underline{b}}$. Thus, all the four standard reverse operators can be implemented.

It may be noted that the circuits were designed essentially to take care of the top half of Table 1 for each of the reverse operators, but they are equally valid for the bottom half. This is because of the reason we had mentioned in Sec. III-2, viz, that the doubtful state can be considered as the Boolean addition of the states $\underline{\underline{T}}$ and $\underline{\underline{F}}$, i.e., $\underline{\underline{D}} = \underline{\underline{T}} \oplus \underline{\underline{F}}$. However, as mentioned earlier in this section, the implementation of the reverse connectives described above, do not take care of the $\underline{\underline{X}}$ input, and the $\underline{\underline{X}}$-check circuit should be suitably used for every one of them, in case the input can have the impossible state $\underline{\underline{X}}$.

..54

3. Unary "and" and unary "or".

We have defined in SNS logic the operators "unary and" and "unary or", for which the equations

$$\underline{\underline{a}}\ A\ \underline{\underline{b}} = \text{true}, \quad \text{and} \quad \underline{\underline{a}}\ O\ \underline{\underline{b}} = \text{true} \tag{13}$$

give the relations between $\underline{\underline{a}}$ and $\underline{\underline{b}}$ as

$$\underline{\underline{a}}\ A\!\rightarrow = \underline{\underline{b}} \quad \text{and} \quad \underline{\underline{a}}\ O\!\rightarrow = \underline{\underline{b}} \tag{14}$$

respectively. The circuits for $A\!\rightarrow$ and $O\!\rightarrow$ have been worked out in order to give the outputs as indicated by Table 2. They are

Table 2. Truth table of the unary connectives $A\!\rightarrow$ and $O\!\rightarrow$.

shown in Fig. 16(a) and (b). The circuit for unary "and" is extremely simple, while that for unary "or" is closely similar to that for unary "implies".

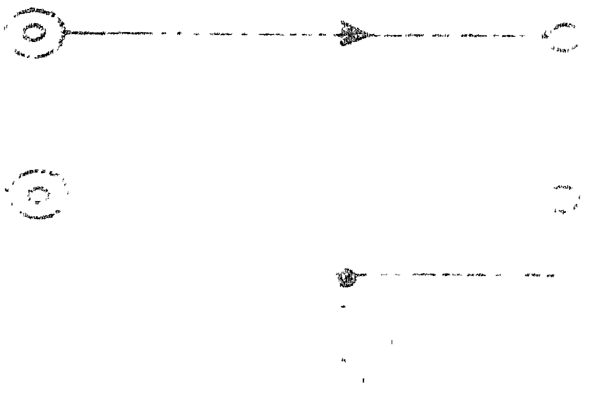
..55

The layout of the modules for the four reverse operators are similar, and are typically represented by $\overleftarrow{O}1$ (for reverse

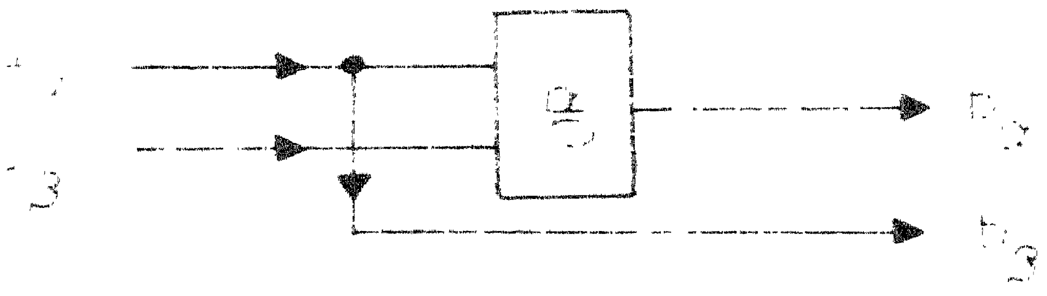
Figure 16. Schematic diagrams of the connectives (a) Unary and $(A\cdot\to)$, (b) Unary or $(O\cdot\to)$.

"or") and $\overleftarrow{A}1$ (for reverse "and") in Panel IV. The layout for the unary A and Unary O are found at the right-hand end of Panel III. The unary "or" is not expected to give any "impossible" signal, and hence it has no yellow lamp, but the unary "and" module has a yellow lamp, as it can give rise to an impossible state under certain circumstances.

..56



[illegible]



[illegible]

Fig. 16

VI. Multiple Operators

1. Multiple "and" and "or"

These two connectives, multiple "and" and multiple "or" are defined by the equations

$$\underline{\underline{a1}}\ A\ \underline{\underline{a2}}\ A\ .\ .\ .\ .\ A\ \underline{\underline{an}} = \underline{\underline{x}} \qquad (15)$$

$$\underline{\underline{a1}}\ O\ \underline{\underline{a2}}\ O\ .\ .\ .\ .\ O\ \underline{\underline{an}} = \underline{\underline{y}} \qquad (16)$$

These can, in principle, be defined as a set of successive operations of binary "and" or binary "or" (See [4]). However, from the point of view of electronics, it is easier to obtain the multiple "and" and multiple "or" implemented via a multiple NAND gate. Such gates are available with 4 inputs and even 8 inputs. In ESNY, 4-input NAND's have been used, and some modules for quadruple "and" and quadruple "or" have been provided. Their circuits are shown in Fig.17 (a) and (b). All that is done is to convert the 4-input NAND into a 4-input OR or a 4-input AND by using inverters, and the circuit is closely similar to what is shown in Fig.10 (a) and (b).

..58

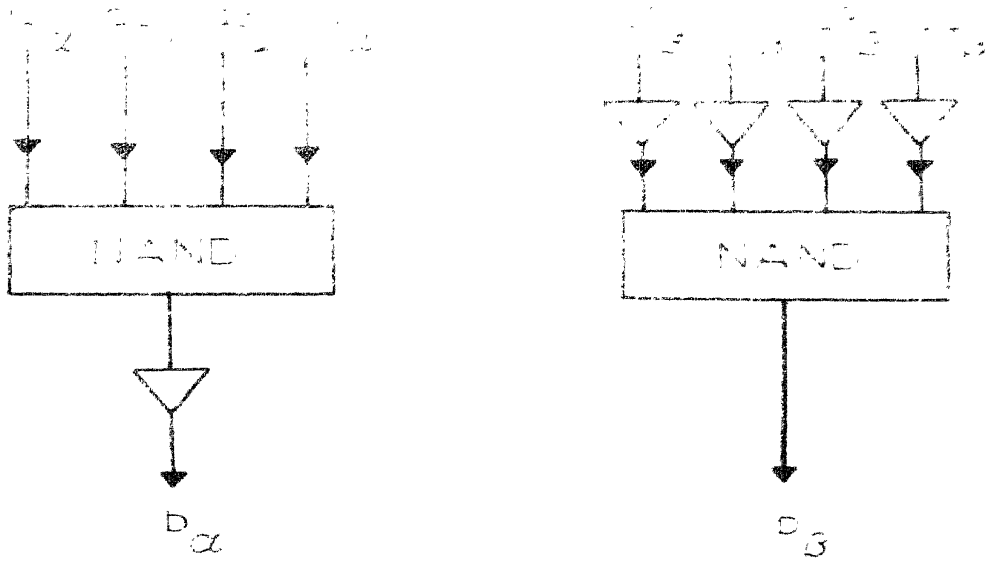
Fig. 17. Schematic diagrams of the connectives (a) multiple and, (b) multiple or

The layout of these modules are shown by A3 and O3 of Panel II. Four input sockets are provided at the left, marked 1, 2, 3 and 4 in Fig.2, and triplicate output sockets are provided. Two additional sockets, marked 5 and 6 are provided at the input side, to facilitate the usage of the module as a 3-input or a 2-input operator. If 5 is connected to 1, it becomes a 3-input module, with 2, 3 and 4 as the inputs. If, in addition, 6 is connected to 4, it becomes a 2-input module, with 2 and 3 as inputs. Both the modules, A3 and O3 are similar in this respect.

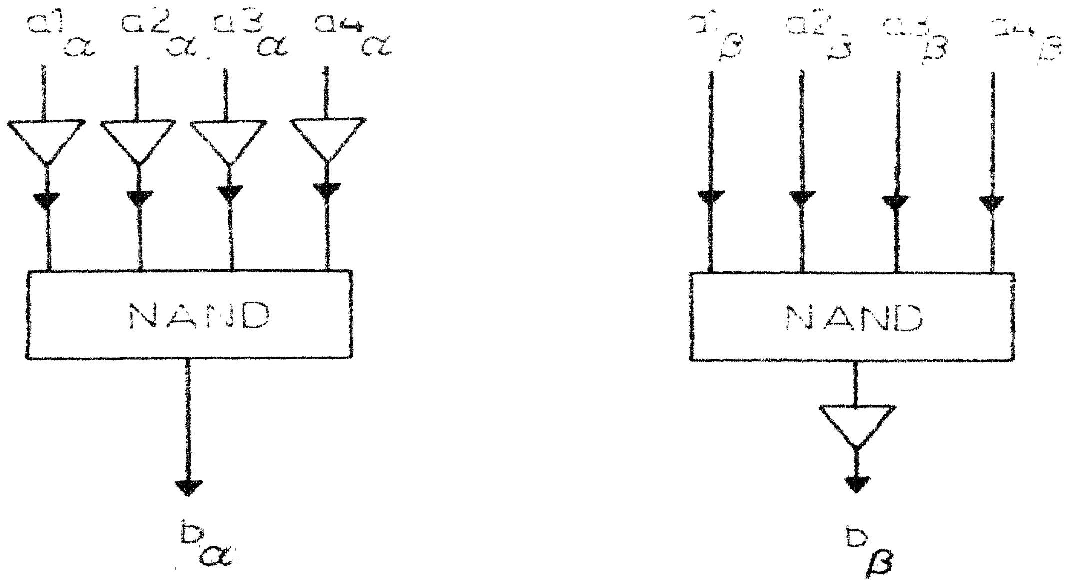
2. Multiple Reverse Operators

The equations (15) and (16) can be reversed in the case when n is 4, to give the term $\underline{\underline{a4}}$ when the inputs are the resultant $\underline{\underline{x}}$ (or $\underline{\underline{y}}$) and the other three inputs are $\underline{\underline{a1}}$, $\underline{\underline{a2}}$ and $\underline{\underline{a3}}$. The

..59



(a)Multiple and



(b)Multiple or

resultant formulae are

$$\underline{\underline{x}}\ \overleftarrow{A}\ \underline{\underline{g}} = \underline{\underline{a4}} = \underline{\underline{b}} \qquad (17a)$$

where

$$\underline{\underline{g}} = \underline{\underline{a1}}\ A\ \underline{\underline{a2}}\ A\ \underline{\underline{a3}} \qquad (17b)$$

In the same way, for 0 also, we have the equation

$$\underline{\underline{y}}\ \overleftarrow{0}\ \underline{\underline{h}} = \underline{\underline{a4}} = \underline{\underline{b}} \qquad (18a)$$

where

$$\underline{\underline{h}} = \underline{\underline{a1}}\ 0\ \underline{\underline{a2}}\ 0\ \underline{\underline{a3}} \qquad (18b)$$

The circuit is then quite obvious. The three inputs $\underline{\underline{a1}}$, $\underline{\underline{a2}}$, $\underline{\underline{a3}}$ marked 1, 2, 3 in the module R02 in Panel IV are connected to a multiple "or" circuit and its output is $\underline{\underline{h}}$. This is put along with $\underline{\underline{c}}$ in a $\overleftarrow{0}$ circuit to obtain $\underline{\underline{b}}$. The inputs $\underline{\underline{a1}}$, $\underline{\underline{a2}}$, $\underline{\underline{a3}}$ are combined in a multiple "and" in the module RA2, to give the output $\underline{\underline{g}}$ (Fig.18) and this is then put along with $\underline{\underline{c}}$ in a $\overleftarrow{A}$ circuit which gives $\underline{\underline{b}}$. In these cases also, two extra sockets 4 and 5 (as marked in Panel IV) have been provided, and if 4 is connected to 1, there are two inputs 2 and 3; if, in addition, 5 is connected to 2, there is only one input of the type $\underline{\underline{a}}$, while the other input, of the type $\underline{\underline{c}}$, is specially marked out.

..61

Fig. 18. Schematic diagram of the "multiple reverse and". (multiple $\overleftarrow{A}$). The corresponding schematic of the multiple $\overleftarrow{O}$ is identical, with O and $\overleftarrow{O}$ in place of A and $\overleftarrow{A}$ respectively.

3. Multiple AY.

For certain purposes, the output of an "and" may be connected to a unary "imply" module Y , in which case we have the operation AY. One module in Panel IV marked A4 is provided for this, and it is incidentally also a multiple-input module. Its circuit is shown in Fig. 19, and no special comments are needed.

Figure 19. Schematic diagram of the module multiple "and implies" (AY).

The circuit for A is shown as a block and the circuit for unary implies Y , is shown as a second block. This module also has two additional sockets at the input-side, for conversion into a 3-input or a 2-input module, as explained in the case of the other multiple-input modules.

..62

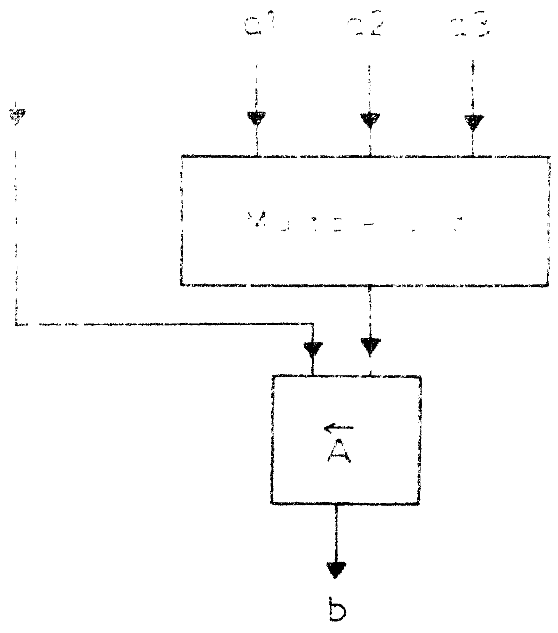
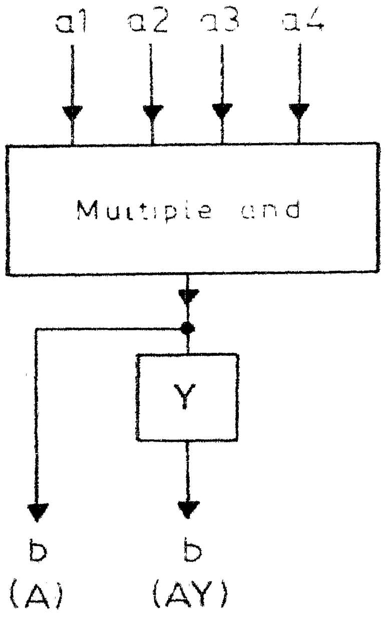


Fig 18.Multiple reverse and



VII. Panel Layout

The panels in ESNY are divided into four parts, viz, Panels I, II, III and IV as mentioned in Sec. I, and they are shown in Figs. 1-4. For ready reference, these are given in a tabular form below in Table 3. The various types of modules are listed here, and the individual modules of each type that occur in the various panels are given in the same line as the name of the module.

Table 3. Details of the modules in ESNY.

When a problem is to be worked out on ESNY, the connections between the various terms, viz, inputs, outputs and inter mediate outputs, are made via the connective modules. For doing this, flexible connectors with input plugs and output plugs are available, having different lengths. They are of two types, as mentioned in Sec. III-1, namely, E-connectors

..64

Table 4.

Nature of module	Names of items present in			
	Panel I	Panel II	Panel III	Panel IV
<u>Input-output</u>				
Input systems	a1, a2	b1, b2	c1, c2	a3
Output systems	x1, x2	y1, y2		x3
Intermediate output	g1, g2	h1, h2		
Input - modifier				e1
<u>Unary modules</u>				
Minus			M1,M2,M3,M4	
Unary "and"			A→	
Unary "or"			O→	
Unary "implies"	Y1, Y2	Y3, Y4		
Unary "implies/implicates"			Y5, Y6	
Unary "not implies"				¥
<u>Binary modules</u>				
Binary "and"	A1, A2			
Binary "or"	O1, O2			
Binary "with"	W1	W2	W3	

Table 3 (Contd....)

Binary "upon"		U1	U2, U3	
Binary "agree"			O1(S),O2(S)	
Binary "similar"		S	S2(O),S3(O)	S4(O)
Binary "imply"				I1
Binary O/I			O4(I)	
Binary ⊖/I			⊖	
Binary "not and"			A	
Binary "reverse and"				RA1
Binary "reverse or"				RO1
Finary "reverse implies"				RI
Binary "reverse implicates"				RJ
Multiple-input modules				
"Multiple and"		A3		
"Multiple or"		O3		
"Multiple and implies"				A4
"Multiple reverse and"				RA2
"Multiple reverse or"				RO2

..66

and N-connectors. An E-connector directly connects the output from one module to the input of the other, while an N-connector interchanges the α- and the β-lines. This is very convenient when a connective N occurs between two modules. For example, if we wish to have ($\underline{\underline{a}}$ N)Y, then the input system $\underline{\underline{a1}}$ is connected via an N-connector to the module Y1, and its output is taken. Similarly, if we wish to have (($\underline{\underline{a1}}$ N) A $\underline{\underline{a2}}$)N, then $\underline{\underline{a1}}$ is connected by an N-connector and $\underline{\underline{a2}}$ by a E-connector to the module A1, and its output is taken via an N-connector. Such instances occur quite often in problems in logic, and the N that occurs is usually implemented by using an N-connector cable, except in rare examples like the modules ~~A~~ and ~~O~~ which themselves give the outputs "AN" and "ON".

It is very important that the output of one module should not be connected to the output of another, as this would lead to shorting. Therefore, the input socket-plug pair, and the

..67

output socket-plug pair are made to be of different sizes, so that are cannot be connected to the other. Also, every input has only one socket for connection, so that two outputs can never be connected to the same input. This simple arrangement has been found to be immensly helpful in always making a safe connection from an output to an input via the E-connectors and the N-connectors, every one of which has an input plug at one end and an output plug at the other end. The input plugs are all of the same colour, viz, black, while the output plug for the E-connector is made black, and the N-connector red to distinguish between the two. This also has been found to be very convenient in actual use. A photograph of the front appearance of ESNY-2 is shown in Fig. 20, with the connections made for a simple problem.

Fig.20. Photograph of the analog computer ESNY2, wired to solve the problem given in Section VIII-2.

..67a

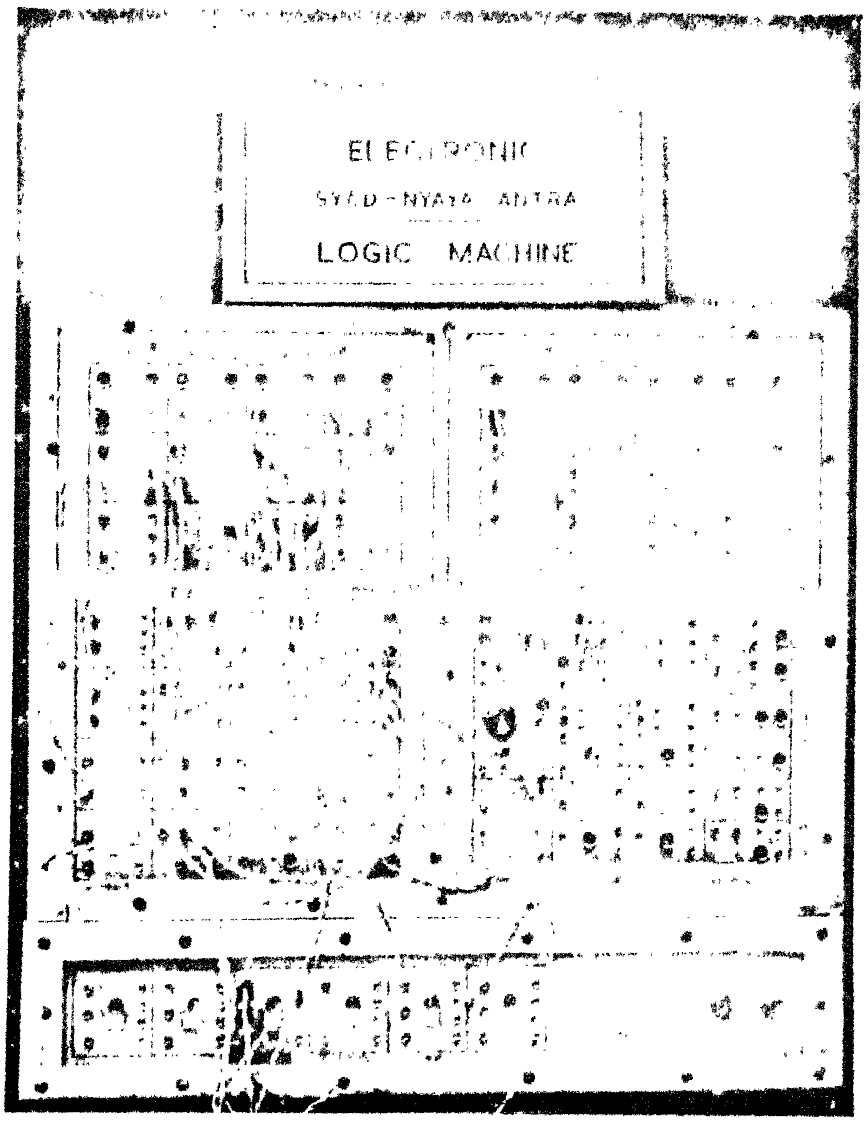


Fig.20. Photograph of the analog computer ESNY2, wired to solve the problem given in Section VIII-2.

VIII. Practical Applications

1. Truth tables.

Since a large variety of connectives of applicability in propositional logic have been implemented in this analog machine, it was employed to work out the truth tables for all inputs — **T**, **F**, **D** in the case of classical connectives, and **T**, **F**, **D**, **X** in the case of forward SNS operators, like W, U, G and M. These tables are listed below. In the case of reverse SNS operators corresponding to the classical ones, only the input states **T**, **F** and **D** are employed, since by the **X**-priority rule (Sec. IV-1), whenever **X** is one of the inputs, the output should also be an **X**.

The truth tables for all the unary connectives are given in two columns against **a** and **b** in the relation **a** Z = **b**. For binary connectives, they are given as 3 x 3 squares (for classical forward and reverse operators) and 4 x 4 squares (for purely SNS operators).

..69

Table 4.

Truth Tables obtained from the ESNY Modules.

(a) Unary Connectives.

a E = b

a	b
T	T
F	F
D	D

a N = b

a	b
T	F
F	T
D	D

a M = b

a	b
T	F
F	T
D	X
X	D

a Y = b

a	b
T	T
F	D
D	D

a V = b

a	b
T	D
F	F
D	D

a A→ = b

a	b
T	T
F	X
D	T

a O→ = b

a	b
T	D
F	T
D	D

(b) Classical binary forward connectives

a A b = c

b a	T	F	D
T	T	F	D
F	F	F	F
D	D	F	D

a O b = c

b a	T	F	D
T	T	T	T
F	T	F	D
D	T	D	D

a I b = c

b a	T	F	D
T	T	F	D
F	T	T	T
D	T	D	D

a S b = c

b a	T	F	D
T	T	F	D
F	F	T	D
D	D	D	D

(e) Classical binary reverse connectives

$c \overleftarrow{A} a = b$

c \ a	T	F	D
T	T	X	T
F	F	D	[illegible]
D	D	D	D

$c \overleftarrow{O} a = b$

c \ a	T	F	D
T	D	T	D
F	X	F	F
D	D	D	D

$c \overleftarrow{I} a = b$

c \ a	T	F	D
T	T	D	D
F	F	X	T
D	D	D	D

$c \overleftarrow{J} a = b$

c \ a	T	F	D
T	D	F	D
F	X	T	T
D	D	D	D

(d) Purely SNS binary connectives

a W b = c

a b	T	F	D	X
T	T	X	T	X
F	X	F	F	X
D	T	F	D	X
X	X	X	X	X

a U b = c

a b	T	F	D	X
T	T	D	D	T
F	D	F	D	F
D	D	D	D	D
X	T	F	D	X

a G b = c

a b	T	F	D	X
T	T	F	F	F
F	F	T	F	F
D	F	F	T	F
X	F	F	F	T

..73

2. Solution of a problem on the effects of voting procedures.

Examples of the application of SNS logic have been given in previous Matphil Reports 8 and 12. Here, we shall give a fresh problem, which is rather long, but which definitely requires the operations and states of SNS logic for its implementation, and explain how ESNY becomes useful for solving it, and for answering various related questions. The problem is as follows:

(a) Problem of the Committee that never met

A Committee was formed consisting of four members - Mr. Trombay (T) and Mr. Colaba (C) from Bombay, and Mr. Peenya (P) and Mr. Jalahalli (J) from Bangalore, to examine the Expansion Programme of the Centre for Environmental Studies in Patna. One year had passed, because no common date could be found for all the four to visit Patna for a meeting. Therefore, the Under-Secretary in Delhi has been told to do the following to get the consolidated recommendation of the Committee by correspondence.

The following rules of procedure were laid down.

1. (i) The two members in Bombay will meet and discuss, and thereafter send their opinions to the Secretary seperately, each

..74

in the form of a vote in a sealed cover — (a) yes, (b) neutral, or (c) no, to stand for the recommendations.

(a) yes $\longmapsto$ Give the grants asked for

(b) neutral $\longmapsto$ Maintain the grants at th present level

(c) no $\longmapsto$ Stop giving any more grants from Delhi.

(ii) The same is to be done by the two members in Bangalore, after a meeting.

(iii) After receiving the four votes, and opening the covers containing them, the Secretary is to take the following steps 2, 3 and 4, to arrive at the final decision.

2. (i) For the Bombay Subcommittee; since both members are senior, the net vote (G) will be "yes" only if both agree to say "yes". If either says "no", the net vote is "no", otherwise "neutral".

(ii) For the Bangalore Subcommittee, it is the reverse, since they are relatively junior. If either says "yes", then the net vote (H) is "yes"; if both say "no", it is "no",

..75

otherwise "neutral".

3. (i) The net votes G and H, from Bombay and Bangalore, are to be again combined by the Secretary in Delhi to give a second stage net vote (V1) following exactly the same procedure as in 2 (ii).

(ii) Also, another second stage vote (V2) is arrived at by the Secretary, considering only the votes sent in by Mr. Trombay and Mr. Peenya, and applying the same procedure as in 2 (i).

4. Having worked out the votes V1 and V2, the Secretary will then check one with the other for "consistency" and arrive at the final vote V as follows.

If both are identical — (a) both "yes", (b) both "neutral", or (c) both "no" — then V is taken to be (a) yes, (b) neutral and (c) no, and action taken, as per (a), (b), (c) of 1 (i).

However, if the two are not identical, and one is neutral, then the other definite vote counts, and either (a), (b), or (c)

..76

of 1 (i) is followed.

On the other hand, if one of V1 and V2 is "yes", and the other "no", the Under-Secretary shall not take any action, but send back the files to his office, for being placed before the Executive Committee one year hence, when it meets, for necessary action.

5. Answer the following questions:

(i) If actually Mr. Trombay said "yes", Mr. Colaba said "no", Mr. Peenya said "yes" and Mr. Jalahalli said "neutral", what action will the Secretary take?

(ii) Is this way of voting "fair" to the Centre for Environmental Studies, in that it is impossible for any one member alone to cast a vote such that the final vote V become a "no", irrespective of the votes cast by the others?

..77

(iii) Conversely, can any one member, or two members, vote in such a way that the final vote V is "yes", irrespective of those of the others?

(b) Implementation on ESNY.

In order to implement this problem on ESNY, we must first write the logical equations in terms of the SNS logical states and operations. Representing the votes of the four persons Mr. Trombay, Mr. Colaba, Mr. Peenya, and Mr. Jalahalli, by the terms $\underline{\underline{t}}$, $\underline{\underline{c}}$, $\underline{\underline{p}}$ and $\underline{\underline{j}}$ respectively, the conditions laid down in the problem can be given by the following logical equations:

$$\underline{\underline{t}} \; A \; \underline{\underline{c}} = \underline{\underline{g}} \qquad (19a)$$

$$\underline{\underline{b}} \; O \; \underline{\underline{j}} = \underline{\underline{h}} \qquad (19b)$$

$$\underline{\underline{g}} \; O \; \underline{\underline{h}} = \underline{\underline{v1}} \qquad (19c)$$

$$\underline{\underline{t}} \; A \; \underline{\underline{p}} = \underline{\underline{v2}} \qquad (19d)$$

$$\underline{\underline{v1}} \; W \; \underline{\underline{v2}} = \underline{\underline{v}} \qquad (19e)$$

..78

where $\underline{\underline{g}}$, $\underline{\underline{h}}$, $\underline{\underline{v1}}$, $\underline{\underline{v2}}$, $\underline{\underline{v}}$ stand for the votes G, H, V1, V2, V, as given in the problem.

The first equation stands for the conditions prescribed in 2(i). It should be mentioned that we are using the state $\underline{\underline{T}}$ to stand for the vote "yes" of an individual, $\underline{\underline{F}}$ for "no", and $\underline{\underline{D}}$ for "neutral". In terms of these, the procedure of obtaining the resultant vote $\underline{\underline{g}}$ is seen to be equivalent to the application of the SNS logical connective "and" as given in the first equation. Similarly, the second equation giving $\underline{\underline{h}}$ represents the procedure in para 2(ii) which is seen to be equivalent to the effect of the connective "or" in $\underline{\underline{p}}\ O\ \underline{\underline{j}}$. Since the next procedure in 3(i) is the same as in 2(ii), the same connective "or" is used, and it is represented by the third equation $\underline{\underline{g}}\ O\ \underline{\underline{h}} = \underline{\underline{v1}}$. On the other hand, the procedure stipulated for 3(ii) is the same as in 2(i), and hence the fourth equation employs the connective "and", to give $\underline{\underline{t}}\ A\ \underline{\underline{p}} = \underline{\underline{v2}}$.

..79

Now we come to the last stage of getting the final vote v, from v1 and v2. It is readily verified that the necessary logical operator that has to be used for this purpose is the SNS operator with" (W). This is because it gives "yes", "no" and "neutral" when both v1 and v2 agree. Correspondingly, action is taken as per 1(a), (b), (c). However, if one says "yes" while the other is "neutral" (D), then T counts and similarly for "no" - F prevailing over "neutral" (D). When, however, one is T and the other is F, then we get for v1 W v2 the fourth state, contradiction (X). This leads to a fourth type of action, which we shall denote by "defer". It is readily seen that the truth table for W, given in Table 4 (d), completely fits the the conditions put in para 4, where the two input votes v1 and v2 are taken to be either T, F, or D, according to the answer "yes", "no" or "neutral", while for the output v, T means "give extra grant", F means "stop grant", D means "maintain status quo", while X means "defer".

..80

With the representation of the symbols $\underline{\underline{T}}$, $\underline{\underline{F}}$, $\underline{\underline{D}}$, $\underline{\underline{X}}$ as described above, the nature of the states of $\underline{\underline{g}}$, $\underline{\underline{h}}$, $\underline{\underline{v1}}$, $\underline{\underline{v2}}$ and finally that of $\underline{\underline{v}}$ can be readily be found out for any combination of states (votes) of $\underline{\underline{t}}$, $\underline{\underline{c}}$, $\underline{\underline{p}}$ and $\underline{\underline{j}}$, by making use of ESNY. The relevant logical graph of the problem is shown in Fig. 21, where the equations (19 a to e) are represented. It is only necessary to set $\underline{\underline{t}}$, $\underline{\underline{c}}$, $\underline{\underline{p}}$, $\underline{\underline{j}}$ at different combinations of $\underline{\underline{T}}$, $\underline{\underline{F}}$, and $\underline{\underline{D}}$ states to find out if $\underline{\underline{v}}$ is a $\underline{\underline{T}}$, $\underline{\underline{F}}$, $\underline{\underline{D}}$, or $\underline{\underline{X}}$.

This can be done for all 3^4 (= 81) combinations of the four input states $\underline{\underline{t}}$, $\underline{\underline{c}}$, $\underline{\underline{p}}$ and $\underline{\underline{j}}$. This has been done, and an illustrative set of 20 examples, selected at random, out of these, is shown in Table 5. The answer to the question in 5(a) is given by the fifth row of this table marked by a star, namely $\underline{\underline{T}}$, i.e. that the fresh grants asked for **will** be given.

..81

Figure 21. Logical graph for the grant committee problem.

(c) Answer to Question 5(ii)

Now we come to the question asked in para 5(ii) of the problem, namely whether it is possible for any one member alone to cast a vote, such that the final vote is a rejection of the grant, (i.e. $\underline{\underline{v}} = \underline{\underline{F}}$), irrespective of the votes cast by the others. In principle, this could be answered by carefully looking through the full 81 line table of the type of Table 5. However, this is the type of problem for which the modular arrangement of ESNY is particularly fitted. Thus, the condition stated in para 5(ii) can be checked by using, the connective G, indicated at the top right hand corner of Fig. 21.

Suppose we wish to know if a "no" vote ($\underline{\underline{F}}$) of $\underline{\underline{t}}$ will make $\underline{\underline{v}}$ to be always $\underline{\underline{F}}$, independent of the states of $\underline{\underline{c}}$, $\underline{\underline{p}}$ and $\underline{\underline{j}}$. For this, we make $\underline{\underline{t}} = \underline{\underline{F}}$, and make the set $\underline{\underline{c}}$, $\underline{\underline{p}}$, $\underline{\underline{j}}$ successively go

..82

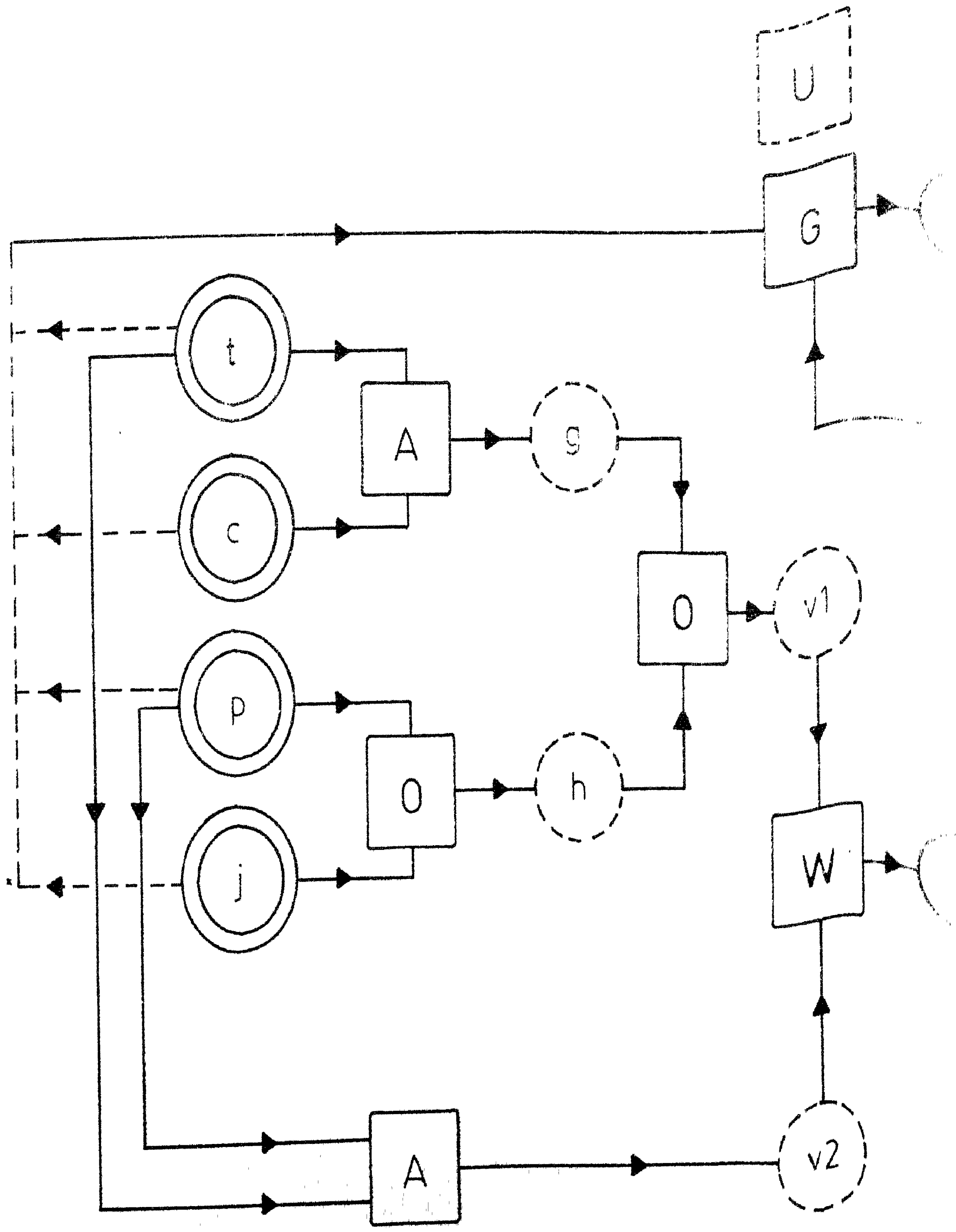


Fig.21

over the 3^3 (= 27) possible combinations. The output of the modules t and v are connected to the G module, so that the final

---

Table 5. A representative set of results for the final vote in the grant committee problem.

---

output x will be T (indicated by a green lamp) only if v is F.

When this test was made, x was not uniformly green, and for some combinations of c, p, j, the final vote was not a 'reject", namely F. The same can be done by connecting c to G, and making c = F, and allowing the other three to take all possible 27 combination of states. (The connections from t, c, p, j to G are shown by dotted lines in Fig.21, to indicate that only one of them is so connected at a time). Once again, the answer was that c alone could not make the final decision v to be a "no". An exactly similar result was found on repeating this process by singling out p alone, or j alone, to be F, and verifying if this led to an F always. It was thus verified that the answer

..84

Table 5

t	c	p	q	e	h	v1	v2	v
T	T	T	T	T	T	T	T	T
T	T	T	F	T	T	T	T	T
T	T	F	F	T	F	T	F	X
T	T	D	T	T	T	T	L	T
T	F	T	D	F	T	T	T	T*
T	F	F	F	F	F	F	F	F
T	D	F	D	D	D	D	F	F
T	D	D	T	D	T	T	D	T
F	T	F	T	F	T	T	F	X
F	F	T	T	F	T	T	F	X
F	F	D	L	F	D	D	F	F
F	D	T	L	F	T	T	F	X
F	D	F	F	F	F	F	F	F
D	T	T	T	L	T	T	D	T
D	T	D	T	D	T	T	D	T
D	F	T	F	F	T	T	D	T
D	F	F	D	F	D	D	F	F
D	F	D	F	F	D	D	D	D
D	D	D	T	D	T	T	D	T
D	L	D	D	D	D	D	D	D

16

to the question in para 5(ii) of the problem is that the voting procedure is definitely "fair", and no single person can dominate the final vote.

(d) Supplementary question to 5(ii)

At this stage, it was felt that we might try these tests with the connective U, instead of G, at the top right hand corner. It will be seen from Table 4(d) that, if one input is F, the output of a binary U is F if and only if, the other input is either F or X. Hence, the same test as above, but repeated with U instead of G, will indicate if the negative vote of one person can make the Under Secretary stop the grant or defer the decision to continue the grant (i.e. it will not even be "status quo" continuation of the grant.). When this test was done, it was found that if either of the senior scientists, t alone, or p alone, voted "no", then the vote v was for grant to be rejected, or the consideration to be deferred, (v = F or X). The 27

..86

possibilities thus obtained for **t** = **F** are given in Table 6. It will be seen from this that some of the final votes are **F**(reject) and some of them are **λ** (defer).

Table 6. Effect of the "no" vote of one member (**t**)

When however, this test was done with **c** = **F**, it was found that all possibilities occured for **v** as shown by the fact that **x** was sometimes red (indicating **F** or **X**) and sometimes both red and green (indicating **T** or **D**). In other words, a negative vote of **c** does not have even the property making the final result to be either "defer" or "reject", all by itself. The same result was obtained with **j** = **F**, as for **c**, but for the negative vote of **p** the effect was the same as for **t**, namely that it always led to a "defer" or "reject".

..87

Table 6

$\underline{\underline{t}}$	$\underline{\underline{c}}$	$\underline{\underline{p}}$	$\underline{\underline{j}}$	$\underline{\underline{v}}$	$\underline{\underline{x}}$
F	T	T	T	X	F
F	T	T	F	X	F
F	T	T	D	X	F
F	T	F	T	X	F
F	T	F	F	F	F
F	T	F	D	F	F
F	T	D	T	X	F
F	T	D	F	F	F
F	T	D	D	F	F
F	F	T	T	X	F
F	F	T	F	X	F
F	F	T	D	X	F
F	F	F	T	X	F
F	F	F	F	F	F
F	F	F	D	F	F
F	F	D	T	X	F
F	F	D	F	F	F
F	F	D	D	F	F
F	D	T	T	X	F
F	D	T	F	X	F
F	D	T	D	X	F
F	D	F	T	X	F
F	D	F	F	F	F
F	D	F	D	F	F
F	D	D	T	X	F
F	D	D	F	F	F
F	D	D	D	F	F

(e) Answer to question 5(iii)

In order to answer this question, the same experiment was tried using G again, but putting t, c, p and g as "yes" successively, and varying the states of the others in each case. The answer in every case was that no one member could vote 'yes' and make the extra grant to be approved (i.e. make v = T), independent of the others.

Considering the second possibility in 5(iii), viz, whether two members can vote in such a way that the final vote is "yes" irrespective of the other two, the trial was made as follows: Consider the case when t and c, both give non-negative votes, i.e. one of them can be T and the other D. However, we have just now seen that if only one is T, this is not sufficient to swing the final vote to be T always. Therefore, we need try only the case when two of the four members both give their vote as "yes". In such a case if the outputs of these two are

..89

connected to an "and" module and its output connected to the G module, with y being the other input, we can then check for y to be always T, by verifying that x is always "green". This was done for all the 6 possible pairs which can be obtained from t, c, p and j. It was found that only for one pair, namely t and p, was the final vote always "yes" (y = T), independent of the votes of the others.

It will be seen from all these that ESNY has a wide range of possibilities, and that it can be adopted to answer various questions relating to a problem connected with logic, as in the present example. All the problems mentioned in [2], [3] and [4], have been put on ESNY, and their solutions could be obtained with very little labour.

3. Convenience and utility of ESNY

The most useful application of this machine will be in the teaching of logic. Where one combines the effect of two connec-

..90

tives, or when one wants to convert a forward relation into reverse relation, it is very easy to make these corresponding connections in the machine, and get the output of the truth table by setting the state of the input to the possibilities $\underline{\underline{T}}$, $\underline{\underline{F}}$ and $\underline{\underline{D}}$ sequentially, using the switches in the input module. Occasionally when needed, we can even have an $\underline{\underline{X}}$-input by taking a $\underline{\underline{D}}$-input and modifying it by using the module M to convert it into the $\underline{\underline{X}}$ state. In fact, complicated truth tables, like those shown in Tables 5 and 6, can be readily read out of the machine (without errors) and the results worked out by hand can be checked against these. Thus, we feel that this machine will be extremely valuable for the teaching of logic.

Similarly, if a problem containing a number of steps has to be solved, the student can perform the solution using the machine in order to check the result that he has obtained. In fact, we have provided enough modules and enough sockets for the outputs of each module, to have a check on the intermediate terms in the

..91

argument as well. Thus, the so-called intermediate input-output modules can be used for monitoring the intermediate terms wherever necessary. We have found by experience that, if the number of inputs is of the order 4 or 5, the number of possibilities of the set of input states is of the order of 3^4 or 3^5 , which is so large that one is likely to commit errors if all the truth tables are worked out manually. Therefore, such complex problems are best examined by using ESNY. Even if computer programs of the type discussed in [3] are available, an interaction with a digital computer is much more difficult than with ESNY, in which switches can be readily set to give different states, and even the problem itself can be modified to correspond to any situation by merely taking out a small number of plugs and inserting them in new places.

We have found this machine to be extremely useful in practice, and we should mention that all the ideas required for producing the FORTRAN program (reported in an earlier Report 12) actually arose out of our experiences with this analog computing machine.

REFERENCES

1. Ramachandran, G.N. "Syad-Nyaya-System Logic -- Essential ideas connected with propositional calculus" -- Matphil Reports No.1, 1979.

2. Ramachandran, G.N. "Computerization of Logic" Golden Jubilee commemoration Volume, National Academy of Science, India, Allahabad, 1980 (in press) (Matphil Reports No. 3).

3. Ramachandran, G.N. and Thanaraj, T.A., "Fortran Program for Sentential Logic", Matphil Reports No. 12, Sep. 1980.

4. Ramachandran, G.N. "Principles of Practical Logic" (under preparation).

F O U R I E R T R A N S F O R M S

A N D

INFINITE DIFFERENTIABILITY OF FUNCTIONS

G.N. Ramachandran and S.H. Kulkarni

Mathematical Philosophy Group
Indian Institute of Science
Bangalore 560 012.

* MATPHIL Reports No. 14, October 1980.

Preface

This report is the outcome of some interesting preliminary studies, on the nature of the differentiability of functions, in relation to the properties of their Fourier Transforms, made by the senior author (GNR). Being a theoretical physicist, he wanted to have these checked by a competent pure mathematician, and he was fortunate enough to obtain the services of the junior author (SHK), who had been trained in functional analysis, for this purpose. Most of the results reported here were obtained during the last three months and all the proofs have been worked out by SHK. These studies were greatly facilitated by having discussions with a pure mathematician (Prof. M.S. Ramanujan), an electrical engineer (Prof. V. Krishnan) and a physicist (Dr. Ramesh Narayanan). We are grateful to all these persons for stimulating discussions.

The idea of "scale invariance", we believe, has been used here for the first time to get a definition of "infinite differentiability" of a function, which is independent of the scale used.

It is interesting to find that this has a close relation to the "moments" of the Fourier transform function. Just as infinite differentiability is defined by requiring that, for $j \longmapsto \infty$, $f^{(j)}(x) \leqslant KC^{j}$, where K, C are finite constants, the existence of moments M_j of all orders j, including $j \longmapsto \infty$ can be defined by requiring that, for $j \longmapsto \infty$, $M_j < K'C'^{j}$, and K', C' are finite. This latter aspect has not been explored further in this report, except where needed to prove the former.

Only functions $f(x) \in L^1(R)$ are discussed in this report. We hope to extend the discussion to square-integrable functions in a further study in due course.

G.N.R.
S.H.K.

FOURIER TRANSFORMS AND INFINITE DIFFERENTIABILITY OF FUNCTIONS

1. Introduction

The ideas of a continuous function and a differentiable function (say, of one real variable) are well known in mathematics [1], [2]. The ideas can be readily extended to any number of differentiations, and one can thus talk about the jth differential coefficient $f^{(j)}(x)$ for any j. It is well known that an analytic function of a complex variable has differential coefficients of all orders [3] and that, in the region in which it is analytic, it has a Taylor expansion of the form

$$f(x+h) = f(x) + hf^{(1)}(x) + \frac{h^2}{2}f^{(2)}(x) + \ldots + \frac{h^j}{j!}f^{(j)}(x) + \ldots\ldots \tag{1}$$

In this equation, the existence of the Taylor expansion - that is, the convergence of the above series - depends on the fact that

$$\lim_{j\to\infty} \frac{f^{(j)}(x)}{j!} = 0 \tag{2}$$

Thus the question arises as to whether $f^{(j)}(x)$ exists for all j, and particularly, whether the limiting value, $\lim_{j\to\infty} f^{(j)}(x)$,

..2

exists. for $\underline{j} \longmapsto \infty$ We may point out that this condition is not necessary for the convergence of the Taylor series as $\underline{j}!$ is in the denominator, so that even if $\underline{f}^{(\underline{j})}(\underline{x}) \longmapsto \infty$ as $\underline{K}\underline{C}^{j}$ where $\underline{K}$, $\underline{C}$ are constants, the Taylor series can still converge.

We were led to a study of values of different orders of differentiation from the study of the relationship between the functions $\underline{f}^{(\underline{j})}(\underline{x})$ and the functions $\underline{u}^{\underline{j}}\underline{g}(\underline{u})$, where $\underline{g}(\underline{u})$ is is the Fourier transform (FT) of $\underline{f}(\underline{x})$ [4]. These two functions (namely $\underline{f}^{(\underline{j})}(\underline{x})$ and $\underline{u}^{\underline{j}}\underline{g}(\underline{u})$) are Fourier transforms of each other (but for some constant) in the case of a large number of functions of physical interest [5], [6], [7]. This connection is stated in almost all tables of Fourier transforms (e.g. [8]), and is used for finding solutions of differential equations. However the precise conditions for which $\underline{f}^{(\underline{j})}(\underline{x})$ and $\underline{u}^{\underline{j}}\underline{g}(\underline{u})$ exist for all values of $\underline{j}$ (including ∞) does not seem to have been studied. We have examined this, and in this report, we give conditions on classes of functions for which this holds for all $\underline{j}$ (including ∞).

A number of interesting results about the nature of a function, such as its bounded support, its boundedness, and existence of all derivatives $f^{(j)}$ (including $j \longrightarrow \infty$) etc, are found to be closely related to corresponding properties of its FT. Thus, various types of behaviour of functions ($f(x)$) in relation to the nature of their FT($g(u)$) are also discussed in this report.

2. Definition of Infinite Differentiability

Suppose we take a function $f(x)$ and find the derivatives $f^{(1)}(x)$, $f^{(2)}(x)$, . . ., $f^{(j)}(x)$. We shall now show that the manner in which $f^{(j)}(x)$ varies with j depends on the choice of 'scale' used for x. This is an interesting point of view, and since it is not treated in text books on Calculus, we shall discuss it in some detail.

Consider the function $y = f(x) = \sin x$, shown in Fig.1, and let the unit of length be 1 cm. (marked by 1, 2, 3, . . .) Then its derivative is $f^{(1)}(x) = \cos x$. Suppose we change the

..4

Fig.1. Effect of scale factor on the mathematical representation of the same point function.

scale of $\underline{x}$, and take the unit to be 2 cm, and designate the lengths measured along the $\underline{x}$-axis in this new scale by $\underline{x}'$. These are marked 1', 2', 3', ... in Fig.1. Then, the same function $\underline{f}(\underline{x})$, but using the new scale, becomes

$$\underline{f}(\underline{x}) = \sin \underline{x} = \sin (2\underline{x}') = \phi(\underline{x}') \qquad (3a)$$

Consequently, its first derivative is

$$\phi^{(1)}(\underline{x}') = 2 \cos (2\underline{x}') = 2 \cos (\underline{x}) \qquad (3b)$$

In this equation, the quantity (or number), given by $\cos 2\underline{x}'$ or $\sin (2\underline{x}')$, is exactly the same as that given by $\cos \underline{x}$, or $\sin \underline{x}$, as the case may be, so that we obtain the relation;

$$\left[\underline{f}(\underline{x}) = \phi(\underline{x}'), \text{ with } \underline{x} = 2\underline{x}' \right] \text{ corresponds to } \left[\phi^{(1)}(\underline{x}') = 2\underline{f}^{(1)}(\underline{x}) \right] \qquad (4a)$$

.4a.

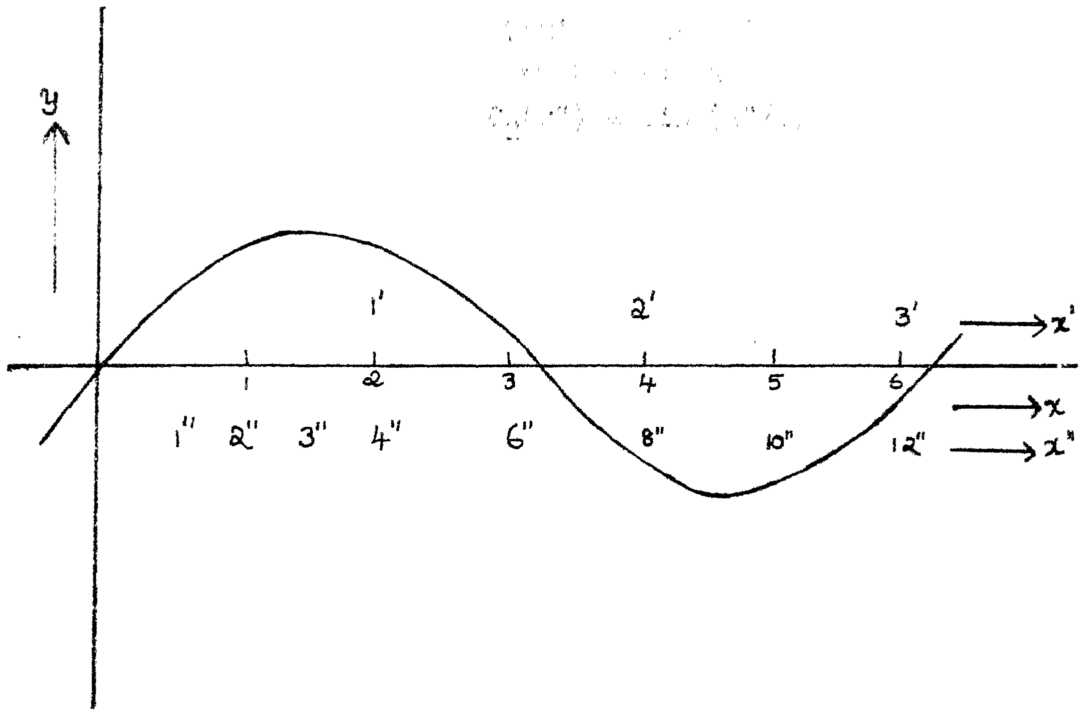


Fig.4. [illegible]
of the same point function.

In the same way, it follows that

$$\phi^{(j)}(x') = 2^{j} f^{(j)}(x) \tag{4b}$$

Therefore, since $|\sin x|$ or $|\cos x|$ is $O(1)$, it follows that

$$f^{(j)}(x) = O(1), \text{ for all } x, \text{ all } j \tag{5a}$$

while

$$\lim_{j \to \infty} \phi^{(j)}(x) = \lim_{j \to \infty} 2^{j}\, O(1) = \infty \tag{5b}$$

In the same way, if we take the unit to be 0.5 cm, and the length along the x-axis, as measured this way, are denoted by $x'' = 1'', 2'', 3'', \ldots$ in Fig.1, we obviously have another description of the same function $f(x)$, in the form,

$$\phi_2(x'') = \sin(x''/2) \tag{6a}$$

and

$$\phi_2^{(1)}(x'') = \tfrac{1}{2}\sin(x''/2) = \tfrac{1}{2}\sin x \tag{6b}$$

Hence, we now find that

$$\lim_{j \to \infty} \phi_2^{(j)}(x'') = \lim_{j \to \infty} \frac{O(1)}{2^{j}} = 0 \tag{6c}$$

Thus, the limiting value of $f^{(j)}(x)$ as $j \longmapsto \infty$ depends on the scale used, and the derivative of order infinity can be either

..6

zero, finite and bounded, or unbounded (infinite), depending on the choice of the scale used for measuring the length $\underline{x}$. More generally, if the unit of $\underline{x}$ is made $\underline{k}$ times the one used for defining $\underline{f}(\underline{x})$, then the 'length' $\underline{x}'$, measured using this unit becomes $\underline{x}/\underline{k}$, so that the same function $\underline{f}(\underline{x})$, expressed in terms of $\underline{x}'$, becomes

$$\phi(\underline{x}') = \underline{f}(\underline{kx}') = \underline{f}(\underline{x}) \qquad (7a)$$

and the derivative $\phi^{(1)}(\underline{x}')$ of $\phi(\underline{x}')$ becomes equal to $\underline{kf}^{(1)}(\underline{x})$.

Consequently, analogous to (4b), we obtain that

$$\phi^{(\underline{j})}(\underline{x}') = \underline{k}^{\underline{j}}\,\underline{f}^{(\underline{j})}(\underline{x}) \qquad (7b)$$

so that, even if $\underline{f}^{(\underline{j})}(\underline{x})$ tends to a finite limit as $\underline{j} \longrightarrow \infty$, $\phi^{(\underline{j})}(\underline{x})$ tends to zero, or infinity, according to the choice of $\underline{k}$ (namely $\underline{k} < 1$, or $\underline{k} > 1$, respectively). Consequently, the statement that "all derivatives exist for a function $\underline{f}(\underline{x})$", is, in general, not independent of the choice of the scale factor, $\underline{k}$. In order to make it independent, we shall require $\underline{f}^{(\underline{j})}(\underline{x})/\underline{KC}^{\underline{j}}$, where $\underline{K}$ and $\underline{C}$ are finite constants, to tend to a finite limit, in the definition

..7

of infinite differentiability. If this happens, then by suitably changing the scale factor, (say making $k > C$) the same function can be made to have 'the derivative of order infinity' to be equal to zero. Hence, we shall specifically define infinite differentiability as follows, so that the definition does not depend on the choice of any particular scale for defining the function $f(x)$.

<u>Definition 1.</u>

A function $f(x)$ is said to be infinitely differentiable, if it has derivatives of all orders and if there exist positive constants K and C such that $|f^{(j)}(x)| \leqslant KC^{j}$, for all j and for all x where $f(x)$ is defined.

When this happens, i.e. $|f^{(j)}(x)| \leqslant KC^{j}$ as $j \longmapsto \infty$,we can change the scale by a factor $1/k$ (the new unit being $(1/k)$ times the old unit), and then, if $k > C$, it follows that $\phi^{(j)}(x') \longmapsto 0$ as $j \longmapsto \infty$. In other words, an infinitely differentiable function (according to Defn.1) can always be reformulated, so that the derivative of order ∞ becomes vanishingly small.

..8

In the discussions that follow regarding the conditions under which a function is infinitely differentiable, we shall find that this question is closely related to the behaviour of the Fourier series (FS), or Fourier transform (FT), of the function under consideration. The two functions $\underline{f}(\underline{x})$ and $\underline{g}(\underline{u})$, which are mutually Fourier transforms of each other, are related to one another by a series of relations of this type, the differential coefficients of one (say $\underline{f}^{(\underline{j})}(\underline{x})$) being representable by integrals involving the function $\underline{u}^{\underline{j}}\,\underline{g}(\underline{u})$ of the FT.

The nature of the FT of functions that are absolutely integrable (that is $\underline{f} \in L^1(R)$) has been studied extensively ([4], [5], [6]). If f is square integrable (that is $\underline{f} \in L^2(\mathbb{R})$), then the FT of $\underline{f}$ is also square integrable, and there is a symmetry between a function and its Fourier transform. However, there are many situations in physics and engineering where absolute integrability is particularly relevant. Hence, this report is devoted to the case of absolutely integrable functions. Obviously, if $\underline{f}$ is

..9

absolutely integrable, it is not necessary that its Fourier transform is also absolutely integrable. Consequently, the results are not symmetric for a function and its FT. ~~In the last section, we shall indicate some possibilities in the case of square integrable functions.~~

3. Fourier Transform and Scale Factor

We shall start by giving the definition of the Fourier transform of a function, without specifying the detailed conditions under which these formulae are valid.

Definition 2.

The Fourier transformation from $f(x)$ to its FT $g(u)$, and the inverse process from $g(u)$ to $f(x)$, are defined as follows:

$$g(u) = \int_{-\infty}^{+\infty} f(x)\, e^{2\pi i u x}\, dx \qquad (8a)$$

$$f(x) = \int_{-\infty}^{+\infty} g(u)\, e^{-2\pi i x u}\, du \qquad (8b)$$

The use of 2π in the exponent agrees with the use of FT in all problems in physics and engineering, $1/u$ ($= \lambda$ say) being

..10

the relevant units of the space associated with $\underline{u}$, since the unit for $\underline{u}$ in (4) is closely related to that of $\underline{x}$ in [illegible]. Thus, if $\underline{x}$ is measured in cm, $\underline{u}$ will be measured in cm^{-1}, with the products of the units of $\underline{x}$ and $\underline{u}$ equal to 1, a dimension-less number. (The word "dimension" is used here in its sense in physics - length has dimension L, velocity LT^{-1}, Fourier space "length" L^{-1}, and so on.) Consequently, we may talk of the Fourier space of $\underline{u}$ as the "reciprocal" space (reciprocal to $\underline{x}$), with the unit of length being taken as cm^{-1}. Obviously, if $\underline{x}$ (and λ) are measured in Å, as in x-ray crystallography, the unit of $\underline{u}$ will be Å^{-1}.

The idea of reciprocal space can be extended to 2 and 3 dimensional spaces (the word "dimension" here has the same connotation as it is normally used in mathematics), and is widely used in ~~xeray~~ crystallography, where FT's in 2 and 3 dimensions occur commonly. (For a discussion of the use of the reciprocal space for describing FT's in such cases, see [9]).

In fact, the equations (8a) and (8b) are not the most general forms that can be adopted for Fourier transformation. Thus, in quantum mechanics, we have the equations

$$\phi(\underline{p}) = \frac{1}{\sqrt{\underline{h}}} \int_{-\infty}^{+\infty} \psi(\underline{x})\, e^{2\pi i \underline{px}/\underline{h}}\, \underline{dx} \qquad (10a)$$

$$\psi(\underline{x}) = \frac{1}{\sqrt{\underline{h}}} \int_{-\infty}^{+\infty} \phi(\underline{p})\, e^{-2\pi i \underline{xp}/\underline{h}}\, \underline{dp} \qquad (10b)$$

and the constant h which occurs therein plays a fundamental role in the theory of quantum mechanics. In fact, the wavelength (λ) of the wave associated with $\underline{p}$ is given by the de Broglie relation

$$\underline{p} = \underline{h}/\lambda \quad , \text{ or } \quad \lambda = \underline{h}/\underline{p} \qquad (10c)$$

and this is responsible for the occurrence of $\underline{h}$ in these equations Hence, if we use the variable $\underline{u}$ to represent $1/\lambda$, then (10a) and (10b) reduce to (8a) and (8b), with $\psi(\underline{x})$ being replaced by $\underline{f}(\underline{x})$, and $\phi(\underline{p})$ by the equivalent $\underline{g}(\underline{u})$ ($= \underline{g}(\lambda)$).

Consequently, by redefining the physical variables, one can always bring any type of formula for Fourier transformation including the one using wave vectors as in (11a) and (11b):

..12

$$\underline{g}(\underline{k}) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{+\infty} \underline{f}(\underline{x})\, e^{\underline{ikx}}\, \underline{dx} \qquad (11a)$$

$$\underline{f}(\underline{x}) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{+\infty} \underline{g}(\underline{k})\, e^{-\underline{ikx}}\, \underline{dk} \qquad (11b)$$

to the standard forms, (8a) and (8b), which we have adopted. Hence, all our formulae in this report will be derived with this particular Definition 2. It is interesting to note that no constant factor occurs in front of the integrals in the formulae (8a) and (8b) for forward and reverse FT's, unlike in the other forms that are employed.

3.1. Scale factor for the Fourier space and jth moments of distribution functions

Suppose we indicate the unit of $\underline{x}$ by $\underline{e}(\underline{x})$, and the corresponding unit of $\underline{u}$ by $\underline{e}(\underline{u})$, so that, in the FT equations (8a) and (8b), we have the relation

$$\underline{e}(\underline{x})\ \underline{e}(\underline{u}) = 1 \qquad (12a)$$

Consequently, if a new unit $\underline{e}'(\underline{x}')$ is taken, equal to $\underline{k}\ \underline{e}(\underline{x})$, then the corresponding new unit for $\underline{u}$ will be $\underline{e}'(\underline{u}')$, given by

$$\underline{e}'(\underline{x}')\ \underline{e}'(\underline{u}') = 1 \qquad (12b)$$

so that

$$\left[\underline{e}'(\underline{x}') = \underline{k}\,\underline{e}(\underline{x})\right] \Longrightarrow \left[\underline{e}'(\underline{u}') = \underline{e}(\underline{u})/\underline{k}\right] \qquad (13)$$

Hence, any change in the choice of the scale factor for $\underline{x}$ also affects the scale used for measuring $\underline{u}$, according to (13).

We shall show in section 5 that $\underline{f}^{(\underline{j})}(\underline{x})$ is closely related to $\underline{u}^{j}\underline{g}(\underline{u})$ by a Fourier transformation given by (14) below, if certain conditions are satisfied.

$$\frac{\underline{d}^{j}\ \underline{f}(\underline{x})}{\underline{dx}^{j}} = \underline{f}^{(\underline{j})}(\underline{x}) = \int_{-\infty}^{+\infty} \underline{u}^{\underline{j}}\, g(\underline{u})\, e^{-\underline{i}2\pi\,\underline{xu}}\,\underline{du} \qquad (14)$$

Hence, if the scale factor is changed as in (13), the value of $\underline{u}^{j}$ is $\underline{k}^{\underline{j}}\,\underline{u}'^{\underline{j}}$ and of $\underline{du} = \underline{k}\ \underline{du}'$. Since $\underline{x}'\underline{u}' = \underline{xu}$ in the exponent (because of (12)), we arrive at the following equations, denoting the function and its FT in the primed scale by $\phi(\underline{u}')$ and $\gamma(\underline{u}')$:

$$\gamma(\underline{u}') = \int_{-\infty}^{+\infty} \phi(\underline{x}')\, e^{2\pi\,\underline{iu}'\underline{x}'}\,\underline{dx}' = \frac{1}{\underline{k}}\,\underline{g}(\underline{u}) \qquad (15a)$$

$$\phi(\underline{x}') = \int_{-\infty}^{+\infty} \gamma(\underline{u}')\, e^{-2\pi\,\underline{iu}'\underline{x}'}\,\underline{du}' = \underline{f}(\underline{x}) \qquad (15b)$$

We thus get back to $\underline{f}(\underline{x})$ by making two FT's in the forward and backward directions, either in the scale $\underline{e}(\underline{x})$, $\underline{e}(\underline{u})$, or in the

..14

scale $\underline{e}'(\underline{x}')$, $\underline{e}'(\underline{u}')$. However, $r(\underline{u}')$ is not identically equal $g(\underline{u})$, but has a further factor $(1/\underline{k})$ multiplying it. [<u>Note</u>: This is because of the definition of $\underline{f}(\underline{x})$ as a "point function ", i.e the values which the function take at the "points" $\underline{x}$, in real space. If we define it as a "density function" (as in probability distribution functions in quantum mechanics and elsewhere, or the electron density function in crystallography), then $\gamma(\underline{u}')$ becomes equal to $g(\underline{u})$ in (15a), with (15b) being unaltered. This question about the distinction between point functions and density functions, in relation to their physical nature, is deferred for a later communication. Here, we take (15a) and (15b) to represent the effects of changing the scale.].

Using (14) in (15 a and b), we obtain that

$$\phi^{(\underline{j})}(\underline{x}) = \int_{-\infty}^{+\infty} \underline{u}'^{\underline{j}}\, \gamma(\underline{u}')\, e^{-2\pi \underline{ix}'\underline{u}'}\, \underline{du}'$$

$$= \underline{k}^{\underline{j}} \int_{-\infty}^{+\infty} \underline{u}^{\underline{j}}\, g(\underline{u})\, e^{-2\pi \underline{ixu}}\, \underline{du} \qquad (16a)$$

which gives the result

$$\phi^{(\underline{j})}(\underline{x}') = \underline{k}^{\underline{j}}\ f^{(\underline{j})}(\underline{x}) \qquad (16b)$$

..15

which is the same as that given in (7), from the definition of the jth derivative itself.

3.2. Use of scale factor to prove infinite differentiability

We shall now use the principle of scale invariance, which we have developed, to show that functions $f(x)$ having certain properties, such as restrictive conditions on the boundedness of $f(x)$, and of bounded support of their Fourier transforms $g(u)$, can be proved to be infinitely differentiable. We are giving one such result here as Theorem 1, even prior to considering general theorems of this nature, in order to indicate how the scale-invariance property can be applied for this purpose.

THEOREM 1

If a function $f(x)$, which may extend from $-\infty$ to $+\infty$ is absolutely integrable and its Fourier transform is of bounded support (i.e. $g(u) = 0$ for $u < -U_1$ and $u > +U_2$, where U_1 and U_2 are finite), then $f(x)$ is infinitely differentiable for all values of x.

We shall give only a brief outline of the proof of this theorem,

..16

as of all the others discussed in this report. Rigorous proofs are being reported elsewhere by one of the authors [10].

Proof of Theorem 1

Since in the definitions (8a) of $g(u)$,

$$|f(x)\, e^{2\pi iux}| \leqslant |f(x)| \qquad (17a)$$

$$|g(u)| \leqslant \int_{-\infty}^{+\infty} |f(x)|\, dx = K, \text{ by assumption} \qquad (17b)$$

Thus, $g(u)$ is bounded, and hence $g(u)\, e^{2\pi ixu}$ is bounded. If, in addition, it is of bounded support, we have

$$|f(x)| = \left| \int_{-U_1}^{+U_2} g(u)\, e^{2\pi iux} du \right| < \int_{-U_1}^{+U_2} g(u)\, du = K' \quad (18)$$

so that $|f(x)|$ is also necessarily bounded.

Now, make a change of scale to $e'(u')$, such that U'_1 and U'_2 are both less than 1. Then, in the u'-picture, we have the result

$$\| u'^{j} \gamma(u') \| = \int_{-U'_1}^{+U_2} | u'^{j} \gamma(u')|\, du' < K'_j \text{ , where } K'_j = \text{a +ve constant} \quad (19a)$$

and further, since $u'^{j} \longrightarrow 0$ for all $U'_1 \leqslant u' < U'_2$,

..17

$$\lim_{\underline{j} \to 0} \| \underline{u}'^{\underline{j}} \gamma(\underline{y}') \| = 0 \qquad (19b)$$

Hence, by (14),

$$|\phi^{(\underline{j})}(\underline{x}')| < \underline{K}' \text{ , a finite + ve number} \qquad (20a)$$

and

$$\lim_{\underline{j} \to \infty} \phi^{(\underline{j})}(\underline{x}') = 0, \text{ for all } \underline{x}' \qquad (20b)$$

Therefore, using the scale $\underline{g}'(\underline{u}')$, the derivatives $\phi^{(\underline{j})}(\underline{x}')$ all tend to zero, as $\underline{j} \to \infty$. Consequently, for any scale,

$$\underline{f}^{(\underline{j})}(\underline{x}) \leqslant \underline{KC}^{\underline{j}} \qquad (21)$$

the required condition for infinite differentiability.

<u>Recapitulation of Theorem 1</u>

We may reiterate the two conditions under which the functions $\underline{f}(\underline{x})$ will be infinitely differentiable:

a) $\underline{f}(\underline{x}) \in L^1(R)$, i.e. it is absolutely integrable

b) FT of $\underline{f}(\underline{x})$ - namely $\underline{g}(\underline{u})$ - is of bounded support.

The most interesting immediate result is that

c) $\underline{f}(\underline{x})$ is bounded.

and then we doduce that

d) $f^{(j)}(x) \leqslant KC^{j}$ i.e. $f(x)$ is infinitely differentiable.

The trick is to find a scale transformation to x', for which

$\phi^{(j)}(x') \longrightarrow 0$ as $j \longrightarrow \infty$.

4. Fourier Series and Infinite Differentiability

4.1. Effect of a finite Fourier series

The subject becomes much more amenable for study if we examine a function which is periodic. We shall take the period to be l, so that $f(x)$ can be written as

$$f(x) = \sum_{-\infty}^{\infty} c_m \, e^{-2\pi imx/l} \tag{22}$$

where Fourier coefficients c_m are given by

$$c_m = \frac{1}{2l}\int_0^l f(x)\, e^{+2\pi imx/l}\, dx \tag{23a}$$

and

$$c_0 = \frac{1}{l}\int_0^l f(x)\, dx \tag{23b}$$

For the case of a Fourier series expansion of a periodic

..19

function, it is possible to prove the following theorem.

THEOREM 2

If $f(x)$ is a periodic function and if it has derivatives of all orders such that $|f^{(j)}(x)| \leq KC^{j}$ for all j and some positive constants K and C, then $f(x)$ has a finite Fourier expansion of the form

$$f(x) = \sum_{-N_1}^{N_2} c_m e^{-2\pi imx/l} \qquad (24)$$

Conversely, if the Fourier expansion of $f(x)$ is finite, the infinite differentiability of $f(x)$ (according to Defn.1) can be derived.

This is also rigorously proved in [10]. Here, we shall demonstrate only the converse by using the technique of choosing a suitable scale factor. Let N be a positive number greater than both N_1 and N_2, and let $2\pi N/l$ be denoted by α. Then choose a new scale $g^*(x')$, with the scale factor $k = 1/\alpha$, so that $x' = \alpha x$. Then,

$$-2\pi imx/l = -i(m/N)\alpha x = -i(m/N)\, x' \qquad (25a)$$

with $N > |m|$. Thus,

..20

$$\phi(\underline{x}') = \sum_{-\underline{N}_1}^{\underline{N}_2} \underline{c}_{\underline{m}} e^{-\underline{i}(\underline{m}/\underline{N})\underline{x}'} \qquad (2$$

where $|\underline{m}/\underline{N}| < 1$. Hence,

$$\frac{d^{\underline{j}}}{dx^{\underline{j}}}\left[e^{-\underline{i}(\underline{m}/\underline{N})\underline{x}'}\right] = (\underline{m}/\underline{N})^{\underline{j}}\ O(1) \qquad (26)$$

so that

$$\phi^{(\underline{j})}(\underline{x}') = \sum_{-\underline{N}_1}^{\underline{N}_2} \underline{c}_{\underline{m}}\ (\underline{m}/\underline{N})\ \ O(1) \qquad (26b)$$

Consequently, since there is a finite number of $\underline{c}_{\underline{m}}$'s ranging from $-\underline{N}_1$ to $\underline{N}_2$ and since for all $\underline{m}$, $(\underline{m}/\underline{N})^{\underline{j}} \longrightarrow 0$ as $\underline{j} \longrightarrow \infty$, it follows that

$$\lim_{\underline{j}\to\infty} \phi^{(\underline{j})}(\underline{x}') = 0 \qquad (27)$$

Thus, we have proved that any function $\underline{f}(\underline{x})$ expressible as a finite Fourier series (FFS) with finite coefficients $\underline{c}_{\underline{m}}$ is infinitely differentiable. The converse of this, namely that infinite differentiability implies an FFS, is shown in [10].

The results given in Corrollary 2.1 below also readily follow

..21

Theorem 2.

COROLLARY 2.1.

If a periodic function $f(x)$ requires an infinite number of terms for its Fourier series representation, then one of the following must be true:

(a) $f^{(j)}(x_0)$ does not exist for some j and x — i.e. $f^{(j-1)}(x)$ is not differentiable; or

(b) $f^{(j)}(x_0) > KC^j$ as $j \longrightarrow \infty$ for any C, i.e. $f(x)$ is not infinitely differentiable.

A simple case requiring an infinite Fourier series

The effective consequence of Theorem 2 is that an infinitely ferentiable function, having a repetitive period l, must essarily be expressible as a sum of a finite number of terms the type $c_m e^{2\pi imx/l}$. It is well-known that an analytic tion of z which is continuously zero (or constant) over a ite range of the variable z, must be zero everywhere. Conse- ntly, it is reasonable to suppose that a periodic function $f(x)$, h is zero over a finite range, x_1 to x_2, of the variable x,

..22

cannot be expressed as a finite Fourier series. We shall give a very elementary proof of this below.

Suppose that $f(x)$ is expressible as the finite Fourier series expansion

$$f(x) = \sum_{-M_1}^{+M_2} c_m e^{-2\pi imx/l} \quad , \quad x_1 < x < x_2 \qquad (28)$$

Multiply both sides by $e^{+2\pi iM_1x/l}$, Then, we can write (28) in the form

$$\sum_{n=0}^{N} c'_n e^{-in\alpha} = e^{2\pi iM_1x/l} f(x) = 0 \text{ ; where } N = M_1 + M_2 \qquad (29a)$$

and

$$c'_n = c_{M_1+m} \; ; \quad \alpha = 2\pi x/l \qquad (29b)$$

If, now, we replace $e^{-i\alpha}$ by δ, Eqn.(29a) is converted into a polynomial equation in δ, of degree N, namely

$$c'_0 + c'_1\delta + c'_2\delta^2 + \ldots\ldots\ldots + c'_N\delta^N = 0 \qquad (30)$$

in which $c'_0, c'_1, \ldots, c'_N$ are given constants, if (28) is true, and $f(x) = 0$ from x_1 to x_2. But we know that a polynomial equation, of finite degree N, can have at most only N roots.

Hence, there can only be N values of ξ, and therefore, N values of x, for which $f(x) = 0$. Since the finite range x_1 to x_2 contains a continuous infinity of points, it follows that no FFS can represent $f(x) = 0$ for this range, unless $f(x) = 0$ for the whole period 0 to l, (and therefore for all x), when all $c_m = 0$.

Obviously, if $f(x) = 0$ for a part (say 0 to l_1) of the period 0 to l, and is non-zero for the rest (l_1 to l) of the period, it is expressed by one analytic function from 0 to l, and by a different one from l_1 to l. Since if a FFS, can represent the function $f(x)$, the function $f(x)$ is necessarily analytic, it follows that it cannot represent both the segments 0 to l_1 and from l_1 to l, in such a case.

Continuing this argument further, even if $f(x)$ is equal to an analytic function $h(x)$ over the whole of one period from 0 to l, if this segment is periodically repeated, it will not be expressible by a FFS, and hence the function $f(x)$ is not infinitely differen-

..24

tiable. (Of course, it is assumed that $\underline{h}(\underline{x})$ itself does no[illegible] a period $\underline{1}$, i.e. it is given by

$$\underline{h}(\underline{x}) = \sum_{-\underline{M}_1}^{+M_2} \underline{c}_{\underline{m}}\, e^{-2\pi i \underline{mx}/\underline{1}} \qquad (31)$$

which begs the question, since (31) is identically equal to (28), and $\underline{f}(\underline{x})$ is then the sum of a FFS).

5. Fourier Transforms of Bounded Support for Non-periodic Functions

5.1. Effect of unbounded support for FT

We start this discussion by referring back to Theorem 1 which states that, for absolutely integrable functions $\underline{f}(\underline{x})$, bounded support of its Fourier transform $\underline{g}(\underline{u})$ will make the function infinitely differentiable. Just as we obtained Corollary 2.1 from Theorem 2 for periodic functions, the following Corollary 1.1, also follows from Theorem 1 [10].

..25

Corollary 1.1

If $\underline{f}(\underline{x})$ is absolutely integrable, and its Fourier transform has unbounded support, then either

(a) $\underline{f}^{(\underline{j})}(\underline{x}_0)$ does not exist for some $\underline{j}$ and $\underline{x}_0$; or

(b) $\lim\limits_{j \to \infty} \underline{f}^{(\underline{j})}(\underline{x}) > \underline{KC}^{\underline{j}}$, for any $\underline{C}$.

The full conditions for this case are described in [10].

Similarly, there is an analogue of the discussion in section 4.2 for periodic functions, as extended to non-periodic functions defined over $-\infty < \underline{x} < \infty$. This is briefly considered below.

5.2. A condition for unbounded support for FT

We showed in section 4.2 that an infinite Fourier series is needed for any periodic function which contains some range ($\underline{x}_1$ to $\underline{x}_2$) over which $\underline{f}(\underline{x})$ is exactly equal to an analytic function $\underline{h}(\underline{x})$ not having this period. We would similarly expect the following theorem to hold for Fourier transforms also.

THEOREM 3

The Fourier transform of any function $\underline{f}(\underline{x})$, which is equal to one analytic function $\underline{h}_1(\underline{x})$ over a part (Δ_1) of its range of $\underline{x}$, say $\underline{x}_1$ to $\underline{x}_2$, and

..26

equal to another analytic function $h_2(x)$ in the remaining part (Δ_2), namely $x = -\infty$ to x_1 and x_2 to $+\infty$, will be of unbounded support.
(Note: This condition is sufficient, but not necessary for the FT to have unbounded support.)

A particularly striking example of this is

$$f(x) = 1 \quad \text{for} \quad -\frac{a}{2} \leqslant x \leqslant \frac{a}{2} \tag{32a}$$

and $$f(x) = 0 \quad \text{for} \quad -\infty < x < -\frac{a}{2}\,,\ +\frac{a}{2} < x < \infty \tag{32b}$$

which is the well-known "rectangular" function, whose FT is the "sinc" function [11]

$$g(u) = \sin(\pi a u)/\pi u \tag{33}$$

for which $g(u)$ exists from $u = -\infty$ to $+\infty$ (unbounded support for the FT).

The sketch of a possible proof of Theorem 3 is as follows [10]. If $g(u)$ is of bounded support, (and is also bounded in magnitude), then

$$p(x) = \int_{-U_1}^{+U_2} g(u)\, e^{-2\pi i x u}\, du \tag{34}$$

..27

is an analytic function of x. Hence, if $p(x) = h_1(x)$ for the finite range Δ_1 of x, the equality should hold for all values of x. Therefore, if $f(x) = h_1(x)$ over Δ_1 and $\neq h_1(x)$ over Δ_2, it cannot be expressed by an integral of the form (34), so that the FT of $f(x)$ is of unbounded support.

It is to be noted that if the range of integration of (34) is from $-\infty$ to $+\infty$, the function $p(x)$ that is obtained is only equal to the original function $f(x)$ almost everywhere, and so it may represent a function of the type $f(x)$ considered in Theorem 3, and be equal to it, except at the points of discontinuity (see[4]).

In physical terms, this result is understandable, for a sharp discontinuity in $f(x)$ or $f^{(j)}(x)$ (for some j) will require waves of all wavelengths λ (= $1/u$), including those with $|\lambda| \longrightarrow \infty$ ($|u| \longrightarrow 0$), for representing it (as is well known).

..28

Corollary 3.1

One consequence of Theorem 3 is that only functions $\underline{f}(\underline{x})$ that are analytic over $\underline{x} = -\infty$ to $+\infty$ can have FT of bounded support. (e.g. $\underline{f}(\underline{x}) = (\sin \underline{x})/\underline{x}$ has as FT the rectangular function). However, the conclusion need not follow for all analytic functions, as in the well-known example of $\underline{f}(\underline{x}) = e^{-\pi \underline{x}^2}$ which has $\underline{g}(\underline{u}) = e^{-\pi \underline{u}^2}$ as its FT, which has unbounded support.

It must be mentioned that the condition, that $\underline{f}(\underline{x})$ having bounded support necessarily leads to $\underline{g}(\underline{u})$ having unbounded support (interchanging $\underline{f}(\underline{x})$ and $\underline{g}(\underline{u})$ in Theorem 1), is not equivalent to the well-known result in FT theory that functions that decay faster than $e^{-\pi \underline{x}^2}$ must have FT decaying slower than $e^{-\pi \underline{u}^2}$. In that result, it is $\sigma(\underline{x})$ and $\sigma(\underline{u})$ that are related by $\sigma(\underline{x})\ \sigma(\underline{u})$ = a constant, and it says nothing about the boundedness or otherwise of the support. Our result may be given a physical interpretation by the following example in quantum mechanics — "A particle restricted to be inside a rectangular well has an

..29

infinite distribution of momentum"; i.e. $-(\underline{a}/2) < \underline{x} < (\underline{a}/2)$ demands that $|\underline{p}|$ can have <u>all</u> values from 0 to ∞, although $|\phi(\underline{p})|$ may tend to zero at $|\underline{p}| \longmapsto \infty$". This is probably the physical origin of tunnelling phenomena observed in all quantum mechanical systems.

<u>5.3. Relation between a function and its FT regarding its infinite differentiability</u>

The above discussion leads to some interesting conclusions about both $\underline{f}(\underline{x})$ and $\underline{g}(\underline{u})$ being of bounded support (BS) and infinitely differentiable (ID). The results are summarized in Table 1, and it assumes that $\underline{f}(\underline{x})$ is bounded in magnitude and absolutely integrable. (Under these conditions, $\underline{f}(\underline{x}) \in L^2(\mathbb{R})$ also, and correspondingly, $\underline{g}(\underline{u}) \in L^2(\mathbb{R})$ also; but it does not follow that $\underline{g}(\underline{u}) \in L^1(\mathbb{R})$.)

..30

Table 1. Nature of Function and FT in Typical Cases

No.	Nature of Support $\underline{f}(\underline{x})$	Nature of Support $\underline{g}(\underline{u})$	Nature of ID $\underline{f}(\underline{x})$	Nature of ID $\underline{g}(\underline{u})$	Comments
1.	Bounded	Unbounded	No	Yes	$\underline{f}(\underline{x})$ = rectangular function $\underline{g}(\underline{u}) = (\sin \pi \underline{u})/\pi \underline{u}$
2.	Unbounded	Bounded	Yes	No	$\underline{f}(\underline{x}) = (\sin^2 \pi \underline{x})/(\pi^2 \underline{x}^2)$ $\underline{g}(\underline{u})$ = triangular function
3.	Unbounded	Unbounded	No	No	$\underline{f}(\underline{x}) = e^{-\pi \underline{x}^2}$ $\underline{g}(\underline{u}) = e^{-\pi \underline{u}^2}$
4.	Bounded	Bounded	-	-	Such a function does not exist.

..31

The three cases are illustrated in Fig.2.

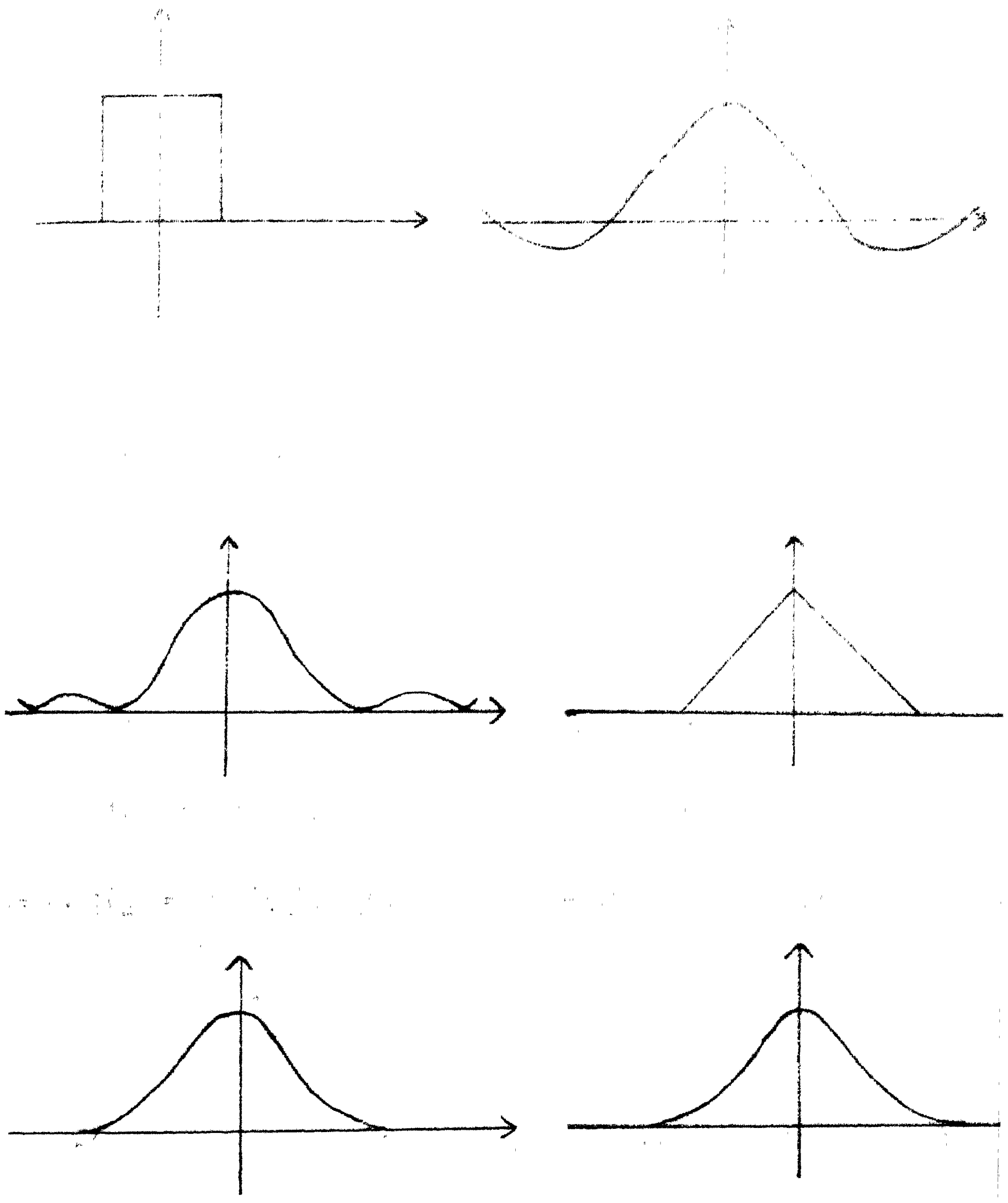
Figure 2. Nature of $\underline{f}(\underline{x})$ and its FT in the three typical Cases 1, 2, 3.

It is interesting that, when both $\underline{f}(\underline{x})$ and $\underline{g}(\underline{u})$ are of unbounded support (Case 3), they are not infinitely differentiable, (although, in the particular example, both are analytic). In fact, all functions of the type $\underline{f}(\underline{x}) = e^{-\pi \underline{x}^2} \underline{H}_{\underline{n}}(\underline{x})$ belong to this class. They are all analytic, having no singularities on the real line, over $-\infty < \underline{x} < +\infty$, but they are not infinitely differentiable (e.g. at $\underline{x} = 0$) ; likewise for their Fourier transforms, namely $\underline{g}(\underline{u}) = i^n e^{-\pi \underline{u}^2} H_{\underline{n}}(\underline{u})$ [6].

6. Concluding Remarks

We thus find that the idea of infinite differentiability of a function $\underline{f}(\underline{x})$ is closely linked with the boundedness or otherwise of the support of its FT. One concludes from this that the only Fourier series, or Fourier transforms, that can represent dis-

..31a.



[illegible]

[illegible]

continuous functions $f(x)$ (or $f^{(j)}(x)$) are those having an infinite number of terms, or of unbounded support, respectively. We also see that, if $g(y)$ has unbounded support, then, even if $f(x)$ is analytic, with no singularities anywhere on the real line, it is not infinitely differentiable.

It is hoped that studies of this type will reveal further new properties of Fourier series and transforms.

7. Acknowledgement

S.H.K. was the recipient of a Research associateship provided to G.N.R. in his project "Studies on Epistemology" under the sponsorship of the University Grants Commission.

..33

References

. Hardy, G.H. 'A Course of Pure Mathematics" Tenth Edn, 1971, Cambridge U. Press, London.

2. Rudin, W. "Principles of Mathematical Analysis", Third Edn, 1976, Mc Graw-Hill Kogakusha, Tokyo.

3. Churchill, R.V. Brown, J.W. and Verhey, R.F. "Complex Variables and Applications", Third Edn, 1974, Mc Graw Hill Kogakusha, Tokyo.

4. Titchmarsh, E.C. "Introduction to the Theory of Fourier Integrals", Second Edn, 1962, Clarendon Press, Oxford.

5. Sneddon, I.A. "Fourier Transforms", 1951, Mc Graw Hill Book Co., New York.

6. Arsac, J. "Fourier Transforms and the Theory of Distributions" 1966, Prentice Hall, Inc, Englewood Cliffs, N.J.

7. Dym, H. and Mc Kean, H.P. "Fourier Series and Integrals" 1972, Academic Press, New York.

8. Spiegel, M.R. "Mathematical Handbook of Formulas and Tables", Schaum's Outline Series, 1968, Mc Graw Hill, New York.

9. Ramachandran, G.N. and Srinivasan, R. "Fourier Methods in Crystallography", 1970, Wiley Interscience Inc, New York.

10. Kulkarni, S.H. (Under preparation).

11. Gaskill, J.D. "Linear Systems, Fourier Transforms and Optics", 1978, John Wiley & Sons, New York.

12. Wiener, N. "The Fourier Integral and Certain of its Applications" 1933, Cambridge University Press, London.

FOURIER ANALYSIS OF INFINITE DIFFERENTIABILITY OF FUNCTIONS

S.H.Kulkarni

Mathematical Philosophy Group

Indian Institute of Science

Bangalore 560 012.

INDIA.

MATPHIL Reports No. 15, January 1981.

Abstract of

"Fourier analysis of infinite differentiability of functions"

by S.H.Kulkarni.

The purpose of this paper is to study "Infinitely differentiable functions" on the real line, defined as those function having all derivatives and satisfying a growth condition, $|f^{(j)}(x)| \leqslant KC^{j}$ for all j and x for some positive constants K and C. The main results obtained are as follows. 1) A periodic function is infinitely differentiable if it has a finite Fourier series expansion. 2) An absolutely integrable (respectively, a square integrable) function on the real line is infinitely differentiable iff the support of its Fourier transform is bounded. 3) Among the absolutely integrable (respectively, square integrable) functions on the real line, the infinitely differentiable functions are precisely the ones, which can be extended to the complex plane as entire functions of the exponential type.

FOURIER ANALYSIS OF INFINITE DIFFERENTIABILITY OF FUNCTIONS

1. INTRODUCTION

It is well known that, if $\hat{f}(u)$ is the Fourier transform of a function $f(x)$, then some constant multiple of $u^j\hat{f}(u)$ is the Fourier transform of $f^{(j)}(x)$, and moreover the differentiability of $f^{(j)}(x)$ is linked with the integrability of $u^j\hat{f}(u)$ [1], [2], [10]. The purpose of this paper is to show that the existence of derivatives of all orders and their growth depends on the nature of the support of the Fourier transform. This result also has an analogue in the case of periodic functions and their Fourier series.

In the second section, we prove that in the case of a periodic function, the existence of derivatives of all orders with a growth condition $|f^{(j)}(x)| \leqslant KC^j$, is equivalent to a finite Fourier series expansion (Theorem 2.1). The analogue of this for absolutely integrable functions is proved in the third section. This states that, in the case of absolutely integrable functions, the same condition (namely existence of derivatives of all orders

.2

with the property $|f^{(j)}(x)| \leq KC^j$) is equivalent to the boundedness of the support of the Fourier transform (Theorem 3.3). This is also shown to be equivalent to a few other conditions. As a consequence, it is shown that these are precisely the functions which can be extended to the complex plane as entire functions of the exponential type (Remark 3.4). Some consequences of this theorem are discussed subsequently. Finally, it is proved that some of these results hold also in the case of square integrable functions on the real line (Theorem 3.10).

We may point out that, for a function $f(x)$, the manner in which the derivatives $f^{(j)}(x)$ behave with respect to j depends on the choice of the scale for x. This can be seen by changing x to x' where $x' = kx$, say. Then the same function $f(x)$ expressed in terms of x' becomes $\phi(x') = f(kx)$ and hence $\phi^{(j)}(x') = k^j f^j (kx)$. Since the techniques involving Fourier series and Fourier transforms are applied to a wide range of problems of of physical interest [1] , [3] , [6] , [8] , [9] , it is useful to study those properties of functions, which are invariant under a change of scale. It can be seen that, the properties studied in this paper (that is, all those statements occurring in Theorems 2.1 and 3.3 as well as the conclusions and corollaries deduced from them) are invariant under a change of scale. The only exception to this is Corollary 3.7.

In view of the above discussion, the following definition of infinite differentiability seems more convenient than the usual one.

DEFINITION 1.1.

A function f(x) is said to be infitely differentiable, if it has derivatives of all orders and there exist positive constants K and C such that $|f^{(j)}(x)| \leq KC^j$, for all j and for all x where it is defined.

The concept of inifinite differentiability, defined in this manner, does not seem to have been studied in the literature, though it seems well-suited for Fourier analysis, as will be shown in this paper. This is so, in spite of the fact that the concept

.4

of quasi-analytic functions [7] and ultradifferentiable functions [4] , which are somewhat close to this concept of infinite differentiability, appear to be known.

A preliminary intuitive account of some of the more important results reported in this paper has been written up as Matphil Reports No.14 from the author's laboratory by G.N.Ramachandran and S.H.Kulkarni. This may be referred to by the more physically oriented readers, for an elementary presentation of the ideas discussed here.

2. ANALYSIS INVOLVING FOURIER SERIES

In this section, using the Fourier series expansion of a periodic function f, we show that finiteness of Fourier series expansion is equivalent to infinite differentiability in the sense of Definition 1.1. For an elementary treatment of the theory of Fourier series, we refer to [2] .

THEOREM 2.1.

Let f be a periodic function on the real line $\mathbb{R}$. Then the following statements are equivalent.

.5

(i) f has derivatives of all orders and there exist positive constants K and C, such that

$|f^{(j)}(x)| \leq KC^j$ for all $j = 1, 2, \ldots$, and for all x in $\mathbb{R}$. (that is, f is infinitely differentiable in the sense of Definition 1.1).

(ii) f has a finite Fourier expansion, that is, f is a trigonometric polynomial of the type

$$f(x) = \sum_{-N_1}^{N_2} c_m e^{-2\pi imx/L} \qquad \text{for some } L > 0.$$

<u>Proof</u>: (i) $\Rightarrow$ (ii)

Let the period of f be L, where L > 0. Then f can be exprêssed as

$$f(x) = \sum_{-\infty}^{\infty} c_m e^{-2\pi imx/L} \tag{2.1.1}$$

where the Fourier coefficients c_m are given by

$$c_m = \frac{1}{L} \int_0^L f(x)\, e^{2\pi imx/L}\, dx \tag{2.1.2}$$

Since f is periodic with the period L, $f^{(j)}$ is also periodic with the period L for every j. Let $c_{m,j}$ denote the mth Fourier coefficient of $f^{(j)}$. Then

$$c_{m,j} = \frac{1}{L} \int_0^L f^{(j)}(x)\, e^{2\pi imx/L}\, dx \tag{2.1.3}$$

.6

We shall first establish a relationship between c_m and $c_{m,j}$.

Thus,

$$c_{m,1} = \frac{1}{L} \int_0^L f'(x)\, e^{2\pi imx/L}\, dx$$

$$= \frac{1}{L} \left[f(x)\, e^{2\pi imx/L} \right]_0^L - \frac{1}{L} \int_0 f(x)\, \frac{2\pi im}{L}\, e^{2\pi imx/I}$$

$$= \left(\frac{-2\pi im}{L} \right) c_m$$

Repeating this process j times, we get

$$c_{m,j} = \left(\frac{-2\pi im}{L} \right)^j c_m \quad \text{for all } j. \qquad (2.1.4)$$

On the other hand, from Equation (2.1.3),

$$|c_{m,j}| \leq \frac{1}{L} \int_0^L |f^{(j)}(x)|\, dx$$

$$\leq \frac{1}{L} \int_0^L KC^j\, dx$$

$$= KC^j \quad \text{for all } j = 1, 2, \ldots \qquad (2.1.5)$$

From Equations (2.1.4) and (2.1.5), we get

$$|c_m| \leq K \left(\frac{CL}{2\pi |m|} \right)^j \quad \text{for all } j.$$

But if $|m| > \frac{CL}{2\pi}$, then $\lim\limits_{j \to \infty} \left(\frac{CL}{2\pi |m|} \right)^j = 0.$

.7

Hence $c_m = 0$ for $|m| > \frac{CL}{2\pi}$. This proves (ii).

(ii) $\Longrightarrow$ (i)

Let f be a trigonometric polynomial given by

$$f(x) = \sum_{m=-N_1}^{N_2} c_m e^{-2\pi imx/L}$$

Then $$f^{(j)}(x) = \sum_{-N_1}^{N_2} c_m \left(\frac{-2\pi im}{L}\right)^j e^{-2\pi imx/L}$$

Let $N = \max \{N_1, N_2\}$

Then $$|f^{(j)}(x)| \leq \sum_{-N_1}^{N_2} |c_m| \left(\frac{2\pi|m|}{L}\right)^j$$

$$\leq \sum_{-N_1}^{N_2} |c_m| \left(\frac{2N\pi}{L}\right)^j$$

$$= \left(\frac{2N\pi}{L}\right)^j \sum_{-N_1}^{N_2} |c_m|$$

for all j and all x.

Thus, letting $K = \sum_{-N_1}^{N_2} |c_m| \geq 0$, $C = \frac{2N\pi}{L} \geq 0$, we see that (i) follows.

Q.E.D.

<u>REMARK 2.2.</u>

It follows directly from the above theorem, that if f is a periodic function with an infinite Fourier Series expansion (that is, infinitely many Fourier Coefficients are nonzero), then f is not infinitely differentiable in the sense of Definition 1.1. This means that one of the following statements must be true.

(i) $f^{(j)}$ does not exist for some j (that is, $f^{(j-1)}$ is not differentiable),

(ii) $f^{(j)}$ exist for all $j = 1, 2, \ldots$, but given any positive constants K and C, there exists a natural number j and a point x_0, such that $|f^{(j)}(x_0)| > KC^j$.

COROLLARY 2.3.

Let f be a non-constant periodic function, which is constant over an interval. Then f is not infinitely differentiable in the sense of Definition 1.1.

Proof. In view of Remark 2.2, we have simply to show that f does not have a finite Fourier Series expansion. Let the period

.9

of f be L. (Note that the length of the interval, where f is assumed to be constant must be less than L, otherwise f will be constant everywhere). Let, if possible, f have a finite Fourier Series expansion

$$f(x) = \sum_{-N_1}^{N_2} c_m e^{-2\pi imx/L} \qquad (2.3.1)$$

Let $z = e^{-2\pi imx/L}$. Then Equation (2.3.1) can be written as

$$f(z) = \sum_{-N_1}^{N_2} c_m z^{-m}$$

$$= \underline{c}_{N_1} z^{N_1} + \underline{c}_{N_1+1} z^{N_1-1} + c_0 + \ldots + c_{N_2} z^{-N_2} \qquad (2.3.2)$$

Without loss of generality, we can assume that the constant value assumed by f on an interval is zero. Now consider,

$$g(z) = z^{N_2} f(z)$$

$$= \underline{c}_{N_1} z^{N_1+N_2} + \ldots + c_0 z^{N_2} + \ldots + c_{N_2} \qquad (2.3.3)$$

From Equation (2.3.3), g(z) is a polynomial of degree N_1+N_2, but $g(z) = 0$ over an interval. This is a contradiction.

Q.E.D.

3. ANALYSIS INVOLVING FOURIER TRANSFORMS

Instead of periodic functions, this section deals with the absolutely integrable functions on the real line $\mathbb{R}$. Using the theory of Fourier transforms, we show that, in the case of a continuous and absolutely integrable function f, boundedness of the support of the Fourier transform is equivalent to infinite differentiability in the sense of Definition 1.1. This has a striking similarity to the results in the previous section. Boundedness of the support is also shown to be equivalent to a few other conditions, leading to some interesting consequences. Some of these results are shown to be true also for the square integrable functions on the real line $\mathbb{R}$.

3.1. NOTATIONS AND DEFINITIONS

Notations in this section are similar to those used in [1]. Let $L^1(\mathbb{R})$ or simply L^1 denote the set of absolutely integrable functions on the real line $\mathbb{R}$, that is, functions f such that $\int_{-\infty}^{\infty} |f(x)|\, dx < \infty$. For f in L^1, we define the norm of f as

.11

$$\| f \| = \int_{-\infty}^{\infty} |f(x)| \, dx \tag{3.1.1}$$

For f in L^1, Fourier transform $\hat{f}$ of f is defined as

$$\hat{f}(u) = \int_{-\infty}^{\infty} f(x) \, e^{2\pi iux} \, dx \quad \text{for} \quad u \in \mathbb{R} \tag{3.1.2}$$

For a function g, we define the inverse Fourier transoform $\check{g}$ of g as

$$\check{g}(x) = \int_{-\infty}^{\infty} g(u) \, e^{-2\pi iux} \, du \tag{3.1.3}$$

provided the integral on the right hand side exists. All integrals are assumed to be taken in the sense of Lebesgue. Following the standard conventions, unless we are discussing a continuous function, we shall always mean by a function f, the equivalence class of all those functions which are equal to f almost everywhere.

Following Wiener [11] we also introduce a subset M^1 of L^1 consisting of all continuous functions f for which, for some positive a and real b,

$$\sum_{n=-\infty}^{\infty} \max_{na+b \leq x \leq (n+1)a+b} |f(x)| < \infty \tag{3.1.4}$$

It can be easily seen that if Equation (3.1.4) holds for some positive a and real b, then it holds for any other positive a_1 and real b_1. In fact, if the integral part of a/a_1 (that is, the greatest integer not greater than a/a_1) is μ , then

$$\sum_{n=-\infty}^{\infty} \max_{(na_1+b_1) \leq x \leq (n+1)a_1+b_1} |f(x)| \leq (\mu + 2) \sum_{n=-\infty}^{\infty} \max_{na+b \leq x \leq (n+1)a+b} |f(x)| \tag{3.1.5}$$

This follows from the fact that no interval of the type $[na+b, (n+1)a+b]$ contains parts of more than $\mu+2$ intervals of the type $[ma_1+b_1, (m+1)a_1+b_1]$. Hence in the definition of M^1 we can take a = 1, b = 0 and replace Equation (3.1.4) by

$$\sum_{n=-\infty}^{\infty} \max_{n \leq x \leq n+1} |f(x)| < \infty \tag{3.1.6}$$

Obviously every function in M^1 belongs to L^1. Wiener has discussed many properties of functions in M^1. In particular, he has proved that, if f is in L^1 and if the Fourier transform $\hat{f}$ of f has a bounded support (that is, if $\hat{f}$ vanishes outside a finite interval), then f belongs to M^1. (See [11] p.80).

.13

The results in this section depend crucially on the following Lemma, which is known (See [1], p.214). We include its proof for the sake of completeness.

LEMMA 3.2.

Let f be a continuous function in L^1 and suppose that the Fourier transform $\hat{f}$, of f, vanishes outside an interval $[-A, A]$. Then f is differentiable and

$$f'(x) = \frac{8A}{\pi} \sum_{n=-\infty}^{\infty} \frac{(-1)^n}{(2n+1)^2} f\left(x + \frac{2n+1}{4A}\right) \qquad (3.2.1)$$

<u>Proof</u>. Since f is in L^1, $\hat{f}$ is continuous ([1], [2]). Hence $\hat{f}$ is absolutely integrable on $[-A, A]$, and we can use the inversion formula (Equation (3.1.3)). Since f is also continuous,

$$f(x) = \int_{-A}^{A} \hat{f}(u)\, e^{-2\pi iux}\, du \qquad \text{for all } x. \qquad (3.2.2)$$

Continuity of $\hat{f}(u)$ implies that f is differentiable and

$$f'(x) = \int_{-A}^{A} (-2\pi iu)\, \hat{f}(u)\, e^{-2\pi iux}\, du$$

$$= -\int_{-A}^{A} 2\pi iu\, e^{\pi iu/2A}\, e^{-2\pi iu(x+\frac{1}{4A})}\, \hat{f}(u)\, du \qquad (3.2.3)$$

.14

Now we consider the function $2\pi iu\, e^{\pi iu/2A}$ as a periodic function with the period 2A and replace it in Equation (3.2.3) by its Fourier series expansion

$$2\pi iu\, e^{\pi iu/2A} = \sum_{-\infty}^{\infty} a_n e^{-\pi iu/A} \tag{3.2.4}$$

where

$$a_n = \frac{1}{2A} \int_{-A}^{A} 2\pi iu\, e^{\pi iu/2A}\, e^{\pi inu/A}\, du$$

$$= \frac{-8A}{\pi} \frac{(-1)^n}{(2n+1)^2} \tag{3.2.5}$$

Then Equation (3.2.3) becomes

$$f'(x) = \int_{-A}^{A} \left[\sum_{-\infty}^{\infty} a_n \exp(-\pi inu/A) \right] \hat{f}(u) \exp\left[-2\pi iu(x + \frac{1}{4A})\right] du$$

$$= - \sum_{-\infty}^{\infty} a_n \int_{-A}^{A} \hat{f}(u) \exp\left[-2\pi iu(x + \frac{2n+1}{4A})\right] du$$

$$= \frac{8A}{\pi} \sum_{-\infty}^{\infty} \frac{(-1)^n}{(2n+1)^2} f(x + \frac{2n+1}{4A})$$

In the above computation, interchanging the order of summation and integration is justified, because the series in Equation (3.2.4) converges uniformly as can be seen from Equation (3.2.5).

Q.E.D.

.15

Now we are in a position to prove the main theorem.

THEOREM 3.3

Let f be a function in L^1. Then the following statements are equivalent.

(i) f has derivatives of all orders, and there exist positive constants K and C such that $|f^{(j)}(x)| \leqslant KC^j$ for all x and all j = 1, 2, . . . (That is, f is infinitely differentiable in the sense of Definition 1.1).

(ii) f can be extended to the complex plane $\mathbb{C}$ as an entire function of the exponential type. (This means that there exist positive constants α and β such that $|f(z)| \leqslant \alpha e^{\beta|z|}$.)

(iii) f is continuous and the support of the Fourier transform $\hat{f}$, of f, is bounded.

(iv) f has derivatives of all orders, $f^{(j)}$ belong to M^1 for all j and there exist positive constants K_1 and C_1, such that

$$\sum_{-\infty}^{\infty} \max_{n \leq x \leq n+1} |f^{(j)}(x)| \leqslant K_1 C_1^j$$

.16

(v) f has derivatives of all orders, $f^{(j)}$ belong to L^1 for all j and there exists positive constants K_2 and C_2 such that $\| f^{(j)} \| \leq K_2 C_2^j$.

Proof.

(i) $\Rightarrow$ (ii)

(This proof closely resembles that of Theorem 19.9 of [7].)

Let a be a real number and consider the Taylor series representation of f(x) around a.

$$f(x) = \sum_{n=0}^{\infty} \frac{f^{(n)}(a)}{n!} (x - a)^n \qquad (3.3.1)$$

We claim that the above series converges for all real x. To prove this, we consider the Taylor formula,

$$f(x) = \sum_{j=0}^{n-1} \frac{f^{(j)}(a)}{j!} (x - a)^j + \frac{1}{(n-1)!} \int_a^x (x - t)^{n-1} f^{(n)}(t)\, dt \qquad (3.3.2)$$

which can be obtained by integration by parts.

Now let,

$$R_n = \frac{1}{(n-1)!} \int_a^x (x - t)^{n-1} f^{(n)}(t)\, dt \qquad (3.3.3)$$

.17

Then

$$|R_n| \leqslant \frac{1}{(n-1)!} \; KC^n \; \left| \int_a^x (x - t)^{n-1} \, dt \right|$$

$$\leqslant \frac{KC^n \, |x - a|^n}{n!} \qquad (3.3.4)$$

Hence $\lim\limits_{n \to \infty} R_n = 0$ for any real x and consequently the Taylor series in Equation (3.3.1) converges for any real x. We can now replace x in Equation (3.3.1) by a complex z. Then this new series converges for any complex z. Hence it defines an entire function $F_a(z)$ in the complex plane $\mathbb{C}$. Moreover, $F_a(x) = f(x)$ for x in $\mathbb{R}$. Therefore various functions $F_a(z)$ coincide on $\mathbb{R}$ and hence they coincide everywhere. Thus they form an analytic extension F of f to $\mathbb{C}$. Also,

$$|F(z)| = |F_0(z)| = \left| \sum_{j=0}^{\infty} \frac{f^{(j)}(0)}{j} \; z^j \right|$$

$$\leqslant \sum_{j=0}^{\infty} K \; \frac{C^j \, |z|^j}{j!}$$

$$= K \, e^{C|z|} \qquad (3.3.5)$$

This proves (ii).

.18

(ii) $\Rightarrow$ (iii)

That f is continuous is trivial. The boundedness of the suppor of $\hat{f}$ follows from the Paley-Wiener theorem. (See [2], pp.158-160 where it is shown that this theorem is also valid for absolutely integrable functions).

(iii) $\Rightarrow$ (iv)

Since f is in L^1 and the support of $\hat{f}$ is bounded; f belongs to M^1. (See [11], p.80 for a proof). Let

$$\sum_{n=-\infty}^{\infty} \max_{n \leqslant x \leqslant n+1} |f(x)| = K_1 \tag{3.3.6}$$

Now let the support of $\hat{f}$ be contained in $[-A, A]$. Then, by Lemma 3.2,

$$f'(x) = \frac{8A}{\pi} \sum_{m=-\infty}^{\infty} \frac{(-1)^m}{(2m+1)^2} f\left(x + \frac{2m+1}{4A}\right) \tag{3.3.7}$$

Hence

$$\max_{n \leqslant x \leqslant n+1} |f'(x)| \leqslant \frac{8A}{\pi} \sum_{m=-\infty}^{\infty} \frac{1}{(2m+1)^2} \max_{n \leqslant x \leqslant n+1} \left| f\left(x + \frac{2m+1}{4A}\right)\right| \tag{3.3.8}$$

Thus,

$$\sum_{n=-\infty}^{\infty} \max_{n \leq x \leq n+1} |f'(x)| \leq \frac{8A}{\pi} \sum_{n=-\infty}^{\infty} \sum_{m=-\infty}^{\infty} \frac{1}{(2m+1)^2} \max_{n \leq x \leq n+1} \left|f\left(x + \frac{2m+1}{4A}\right)\right|$$

$$= \frac{8A}{\pi} \sum_{m=-\infty}^{\infty} \frac{1}{(2m+1)^2} \sum_{n=-\infty}^{\infty} \max_{n \leq x \leq n+1} \left|f\left(x + \frac{2m+1}{4A}\right)\right| \qquad (3.3.9)$$

But from Equations (3.1.5) and (3.3.6)

$$\sum_{n=-\infty}^{\infty} \max_{n \leq x \leq n+1} \left|f\left(x + \frac{2m+1}{4A}\right)\right| = \sum_{n=-\infty}^{\infty} \max_{n-\frac{2m+1}{4A} \leq x \leq n+1-\frac{2m+1}{4A}} |f(x)|$$

$$\leq 3 \sum_{n=-\infty}^{\infty} \max_{n \leq x \leq n+1} |f(x)|$$

$$= 3K_1 \qquad (3.3.10)$$

Hence,

$$\sum_{n=-\infty}^{\infty} \max_{n \leq x \leq n+1} |f'(x)| \leq \frac{8A}{\pi} \sum_{m=-\infty}^{\infty} \frac{1}{(2m+1)^2} 3K_1$$

$$= \frac{8A}{\pi} 3K_1 \frac{\pi^2}{4}$$

$$= 6\pi A K_1 \qquad (3.3.11)$$

Similarly, since the Fourier transform of f'(namely — $2\pi iu\hat{f}(u)$) also vanishes outside the interval $[-A, A]$, we can prove,

$$\sum_{n=-\infty}^{\infty} \max_{n \leq x \leq n+1} |f''(x)| \leq (6\pi A) \sum_{n=-\infty}^{\infty} \max_{n \leq x \leq n+1} |f'(x)|$$

$$\leq (6\pi A)^2 K_1$$

.20

Thus, repeating this process j times,

$$\sum_{n=-\infty}^{\infty} \max_{n \leq x \leq n+1} |f^{(j)}(x)| \leq K_1 (6\pi A)^j \qquad (3.3.12)$$

This proves (iv) by taking $C_1 = 6\pi A$.

(iv) $\Rightarrow$ (i)

This is obvious.

Thus, we have so far proved the equivalence of statements (i), (ii), (iii) and (iv). Also it is easy to see that (iv) implies (v). Now we shall prove

(v) $\Rightarrow$ (iii)

For real a, x and x > a, we have

$$f^{(j)}(x) = f^{(j)}(a) + \int_a^x f^{(j+1)}(t)\,dt \qquad (3.3.13)$$

Since $f^{(j)}$ is in L^1, $\int_a^\infty f^{(j+1)}(t)\,dt$ exists and hence $\lim_{x\to\infty} f^{(j)}(x)$ exists, and since $f^{(j)}$ is also in L^1, this limit has to be zero.

Similar argument holds when $x \longrightarrow -\infty$.

Hence $\lim_{x\to\pm\infty} f^{(j)}(x) = \lim_{x\to\pm\infty} f(x) = 0$, for all j. $\qquad (3.3.14)$

Now,

$$\hat{f}'(u) = \int_{-\infty}^{\infty} f'(x)\, e^{2\pi iux}\, dx$$

.21

$$= \Big[f(x)\ e^{2\pi iux}\Big]_{-\infty}^{\infty} - \int_{-\infty}^{\infty} f(x)\ 2\pi iu\ e^{2\pi iux}\, dx$$

$$= (-2\pi iu)\ \hat{f}(u) \quad \text{by Equation (3.3.14)}$$

Thus, repeating this process j times, we get

$$\hat{f^{(j)}}(u) = (-2\pi iu)^j\ \hat{f}(u) \quad \text{for all u.} \tag{3.3.15}$$

But we also know, from Equations (3.1.1) and (3.1.2),

$$|\hat{f^{(j)}}(u)| \leq \| f^{(j)} \| \leq K_2\ C_2^j \quad \text{for all u and j} \tag{3.3.16}$$

From Equations (3.3.15) and (3.3.16), we get

$$|\hat{f}(u)| \leq K_2 \left(\frac{C_2}{2\pi |u|} \right)^j \quad \text{for all u.} \tag{3.3.17}$$

But, if $|u| > \frac{C_2}{2\pi}$, then $\lim_{j \to \infty} \left(\frac{C_2}{2\pi |u|} \right)^j = 0.$

Hence, $\hat{f}(u) = 0$ for all u, such that $|u| > \frac{C_2}{2\pi}$. This implies (iii) and also completes the proof of the theorem.

Q.E.D.

REMARK 3.4

The equivalence of (i) and (ii) gives us a characterization of those absolutely integrable functions which can be extended to

.22

the complex plane as the entire functions of the exponential type. It means that precisely those functions in L^1, which are infinitely differentiable in the sense of Definition 1.1, can be extended to $\mathbb{C}$ as entire functions of the exponential type. Also, equivalence of (ii) and (v) means that if the restriction to $\mathbb{R}$ of an entire function of the exponential type is in L^1, then restrictions to $\mathbb{R}$ of all its derivatives are also in L^1 and in addition they satisf a growth condition $\|f^{(j)}\| \leqslant K_2 C_2^j$ for some positive constants K_2 and C_2.

REMARK 3.5

It is a direct consequence of Theorem 3.3, that if a continuou function f is in L^1 and its Fourier transform $\hat{f}$ has unbounded suppo then f is not infinitely differentiable in the sense of Definition This means that one of the statements in Remark 2.2 must be true.

COROLLARY 3.6.

Suppose a continuous nonzero function f defined on $\mathbb{R}$ has a bounded support. Then f is not infinitely differentiable in the

.23

sense of Definition 1.1.

Proof. This can be proved in many ways. One way is to note that, under the given conditions, the support of $\hat{f}$ must be unbounded (see [2], p.121 for a proof of this) and then to apply Remark 3.5.

Q.E.D.

COROLLARY 3.7.

Suppose f has derivatives of all orders and $f^{(j)}$ are in L^1 for all j. Then either (i) $\lim_{j\to\infty} \|f^{(j)}\| = \infty$, or (ii) $|f^{(j)}(x)| \leq B(f) = \sup \{|f(x)| : x \in \mathbb{R}\}$ for all j and x.

Proof. Consider Equation (3.3.15),

$$\widehat{f^{(j)}}(u) = (-2\pi iu)^j \hat{f}(u) \quad \text{for all } u \text{ in } \mathbb{R}.$$

Hence, if $\hat{f}(u) \neq 0$, for some u, such that $|u| > \frac{1}{2\pi}$, then

$$(2\pi|u|)^j \; |\hat{f}(u)| = |\widehat{f^{(j)}}(u)| \leq \|f^{(j)}\| \quad \text{and thus}$$

$$\lim_{j\to\infty} \| f^{(j)}\| = \infty$$

.24

Otherwise, $f(u) = 0$ for all u such that $|u| > \frac{1}{2\pi}$

Then, by Lemma 3.2,

$$f'(x) = \frac{-4}{\pi^2} \sum_{n=-\infty}^{\infty} \frac{(-1)^n}{(2n+1)^2} f(x + \frac{(2m+1)\pi}{2}) \qquad \text{for all } x.$$

Hence $$|f'(x)| \leq \frac{4}{\pi^2} \sum_{-\infty}^{\infty} \frac{1}{(2n+1)^2} B(f)$$
$$= B(f)$$

Similarly, we can prove,

$$|f''(x)| \leq \sup \{|f'(x)| : x \in \mathbb{R}\}$$
$$\leq B(f) \qquad \text{for all } x.$$

Thus, in general,

$$|f^{(j)}(x)| \leq B(f) \quad \text{for all } x.$$

Q.E.D.

REMARK 3.8

We note that all statements in Theorem 3.3 except statement (ii) are statements about functions on the real line $\mathbb{R}$, without any reference to functions on the complex plane $\mathbb{C}$. Further,

.25

it can be seen from the proof of Theorem 3.3, that the equivalence of statements (iii), (iv) and (v) can be proved without any use of complex function theory. Also, it is possible to prove statement (i) assuming the validity of any one of the statements (iii), (iv) and (v) without any reference to the complex function theory. But when it comes to proving any of these statements assuming statement (i), the proof has to proceed as (i) $\Rightarrow$ (ii), (ii) $\Rightarrow$ (iii) and then (iii) $\Rightarrow$ (iv) or (iii) $\Rightarrow$ (v). In this process, the proof of (ii) $\Rightarrow$ (iii) is a slight modification of the Paley -Wiener theorem [5] whose proofs use some parts of the theory of analytic functions of a complex variable. (See proofs given in [2] and [7] which make use of the Phragmen-Lindelof theorem and the Cauchy theorei respectively). It would be of interest to know whether it is possible to prove (i) $\Rightarrow$ (iii) (or for that matter (i) $\Rightarrow$ (iv), (i) $\Rightarrow$ (v)) without using the theory of functions of a complex variable.

3.9. SQARE INTEGRABLE FUNCTIONS

Now we shall show that some of the properties, proved so far of absolutely integrable functions, are possessed also by square integrable functions. Let $L^2(\mathbb{R})$ or simply L^2 denote the set of all square integrable functions on $\mathbb{R}$, that is, functions f such that $\int_{-\infty}^{\infty} |f(x)|^2 \, dx < \infty$

For f in L^2, we define the norm of f as

$$\|f\|_2 = \left(\int_{-\infty}^{\infty} |f(x)|^2 \, dx \right)^{\frac{1}{2}} \qquad (3.9.1)$$

For f in L^2, the Fourier transform $\hat{f}$ is defined as

$$\hat{f}(u) = \int_{-\infty}^{\infty} f(x) \, e^{2\pi iux} \, dx \quad \text{for} \quad u \in \mathbb{R} \qquad (3.9.2)$$

where the limit in the right hand side is understood in the sense of convergence in L^2. More precisely, for real a, b and a < b, if we define

$\hat{f}_{a,b} = \int_a^b f(x) \, e^{2\pi iux} \, dx$, then Equation (3.9.2) means that

$$\lim_{\substack{b \to \infty \\ a \to -\infty}} \|\hat{f} - \hat{f}_{a,b}\|_2 = 0$$

It is known that if f is in L^2, then such a limit $\hat{f}$ exists, this $\hat{f}$ is also in L^2 and $\|f\|_2 = \|\hat{f}\|_2$. Moveover, if $\hat{f}$ is the

Fourier transform of f, then

$$f(x) = \int_{-\infty}^{\infty} \hat{f}(u)\, e^{-2\pi iux}\, du \text{ for } x \in \mathbb{R}, \qquad (3.9.3)$$

Where again the limit is understood in the sense of convergence in L^2 as described above. (See [1] , [2] , [7] and [10] for the proofs of these facts).

Now we prove a partial analogue of Theorem 3.3 for functions in L^2.

THEOREM 3.10

Let f be a function in L^2. Then the following statements are equivalent.

(i) f has derivatives of all orders and there exist positive constants K and C, such that $|f^{(j)}(x)| \leq KC^j$ for all x and all j. (That is, f is infinitely differentiable in the sense of Definition 1.1).

(ii) f can be extended to the complex plane $\mathbb{C}$ as an entire function of the exponential type.

(iii) f is continuous and the support of the Fourier transfor

$\hat{f}$ of f is bounded.

Proof. (i) $\Rightarrow$ (ii)

This proof is the same as that of (i) $\Rightarrow$ (ii) in Theorem 3.3.

(ii) $\Rightarrow$ (iii)

This follows from the Paley-Weiner theorem, ([2] , [5] , [7]

(iii) $\Rightarrow$ (i)

Let the support of $\hat{f}$ be contained in $[-A, A]$. From

Equation (3.9.3)

$$f(x) = \int_{-A}^{A} \hat{f}(u)\, e^{-2\pi iux}\, du \qquad \text{for } x \in \mathbb{R} \tag{3.10.4}$$

We shall prove that f is differentiable, and

$$f'(x) = \int_{-A}^{A} (-2\pi iu)\, \hat{f}(u)\, e^{-2\pi iux}\, du \tag{3.10.5}$$

Since $e^{-2\pi iux}$ is a differentiable function with the derivative

$-2\pi iu\, e^{-2\pi iux}$; for every real h , we can find θ , such that

$0 \leqslant \theta \leqslant 1$ and

$$e^{-2\pi iu(x+h)} - e^{-2\pi iux} = h(-2\pi iu)\, e^{-2\pi iu(x+\theta h)} \tag{3.10.6}$$

. 29

From Equations (3.10.4) and (3.10.6), we get

$$f(x+h) = f(x) \;=\; \int_{-A}^{A} h(-2\pi iu)\, \hat{f}(u)\, e^{-2\pi iu(x+\theta h)}\, du \qquad (3.10.7)$$

Hence,

$$\left| \frac{f(x+h) - f(x)}{h} - \int_{-A}^{A} (-2\pi iu)\, \hat{f}(u)\, e^{-2\pi iux}\, du \right|$$

$$= \left| \int_{-A}^{A} (-2\pi iu)\, \hat{f}(u)\, (e^{-2\pi iu(x+\theta h)} - e^{-2\pi iux})\, du \right|$$

$$\leq \int_{-A}^{A} 2\pi|u|\; |\hat{f}(u)|\; |e^{-2\pi iux}|\; |e^{-2\pi iu\theta h} - 1|\, du$$

$$\leq \quad 2\pi A \int_{-A}^{A} |\hat{f}(u)|\; |e^{-2\pi iu\theta h} - 1|\, du$$

$$\leq \quad 2\pi A\, \|\hat{f}\|_2 \left(\int_{-A}^{A} |e^{-2\pi iu\theta h} - 1|^2\, du \right)^{\frac{1}{2}} \qquad (3.10.8)$$

The last inequality in Equation (3.10.8) follows from the Schwarz inequality (See Theorems 3.5 and 3.8 of [7]). It is easy to check that

$$\lim_{h \to 0} \int_{-A}^{A} |e^{-2\pi iu\theta h} - 1|^2\, du = 0$$

Hence from Equation (3.10.8), f is differentiable and f' is given by Equation (3.10.5). Similarly, we can prove that the derivatives of all orders of f exist and $f^{(j)}$ is given by

$$f^{(j)}(x) = \int_{-A}^{A} (-2\pi iu)^j\, \hat{f}(u)\, e^{-2\pi iux}\, du \qquad (3.10.9)$$

Hence $\left| f^{(j)}(x) \right| \leq (2\pi A)^j \, \|\hat{f}\|_2$ as above.

Thus (i) follows by taking $K = \|\hat{f}\|_2$ and $C = 2\pi A$.

Q.E.D.

4. ACKNOWLEDGEMENT

Most ideas in this paper were inspired by the numerous discussions that the author had with Professor G.N.Ramachandran, while working as a Research Associate under the guidance of the latter in the scheme "Studies in Epistemology" sponsored by the University Grants Commission of India.

REFERENCES

1. J. Arsac. "Fourier transforms and the theory of distributions", Prentice Hall, Englewood. Cliffs, N.J. 1966.
2. H. Dym and H.P. McKean. "Fourier series and integrals", Academi Press, New York, 1972.
3. J.D.Gaskill. "Linear systems, Fourier transforms and optics", Hohn Wiley and sons, New York, 1978.
4. H.Komatsu. "An analogue of the Cauchy-Kowalersky theorem for ultradifferentiable functions and a division theorem of ultra-distributions as its dual". J. Fac. Sci., Univ. Tokyo, Sect 1, A 26, 239-254, 1979.
5. R.E.A.C. Paley and N. Wiener. "Fourier Transforms in the comple domain" Amer. Math. Soc. Colloquium Publ. 19, New York, 1934.

.31

6. G.N.Ramachandran and R. Srinivasan. "Fourier methods in Crystallography", Wiley interscience, Inc, New York, 1970.

7. W.Rudin. "Real and complex analysis" Tata McGraw Hill, Bombay, New Delhi, 1966.

8. L.A.Shepp and J.B. Kruskal. "Computerised tomography: The new medical X-ray technology". Am. Math. Monthly., 85, 420-438, 1978.

9. I.A.Sneddon. "Fourier transforms", McGraw Hill, New York, 1951.

10. E.C.Titchmarsh. "Introduction to the theory of Fourier integrals", Clarendon Press, Oxford, 1962.

11. N.Wiener. "The Fourier integral and certain of its applications", Cambridge University Press, London, 1933.

LIST OF SYMBOLS

0 (zero)

1 (one), l (ell)

A, L, K, C, K_1, K_2, C_1, C_2

a, b, k, n, m

α β μ

x, x', u, z

f, g, ϕ, $F_a(z)$

$\hat{f}$, $\check{g}$

$f^{(j)}$, $\phi^{(j)}$

C_m, $C_{m,j}$ a_n, R_n

$\mathbb{R}$, $\mathbb{C}$

$L^1(\mathbb{R})$, L^1, $L^2(\mathbb{R})$, L^2, M^1

$B(f)$

$|$ $|$ $\|$ $\|$ $\|$ $\|_2$

$>$ $\geqslant$, $<$, $\leqslant$, $\Longrightarrow$, $\longrightarrow$, ∞ . Σ $\in$

$($ $)$ $\{$ $\}$, $[$ $]$

Proposed running title - Analysis of Infinite Differentiability

Name of the author - S.H.Kulkarni

Mailing address - Mathematical Philosophy Group
Indian Institute of Science
Bangalore 560 012.
INDIA.

P O C K E T   C A L C U L A T O R   P R O G R A M

T I N Y 1

F O R   S E N T E N T I A L   L O G I C

(For implementation on TI 58/59)

Harishankar Ramachandran* and T.A. Thanaraj

Mathematical Philosophy Group
Indian Institute of Science
BANGALORE 560 012

(* On leave from Department of Electrical Engineering, Indian Institute of Technology, Powai, Bombay 400 078)

MATPHIL Reports No. 16, January 1981.

P O C K E T C A L C U L A T O R P R O G R A M

TINY 1

F O R S E N T E N T I A L L O G I C

(For implementation on TI 58/59)

Harishankar Ramachandran* and T.A. Thanaraj

Mathematical Philosophy Group
Indian Institute of Science
Bangalore 560 012.

(* On leave from Department of Electrical Engineering, Indian Institute of Technology, Powai, Bombay 400 078)

MATPHIL Reports No. 16, January 1981.

CONTENTS

Pocket Calculator Program TINY1 for Sentential Logic (For implementation on TI 58/59)

1. INTRODUCTION

In a previous Report No. 12[1] from this laboratory, the method of programming Syad-Nyaya-System (SNS) logic in FORTRAN was described, with special reference to the use of the logical operators EQU, NOT, AND, OR and XOR available in FORTRAN 10 as implemented on DEC 10. The program was completely based on the use of these operators of classical logic (CL) which act on a Boolean variable that can take the two values 1 and 0, corresponding respectively to the logical states TRUE and FALSE of a term. Using the symbols CA, CB, CC for the Boolean variables representing the terms $\underline{a}$, $\underline{b}$, $\underline{c}$ of classical logic, the logical equations and FORTRAN statements that correspond to these are as follows:

$\underline{a} = \underline{b}$;   CB = CA   (1a)

$\neg\underline{a} = \underline{b}$;   CB = .NOT.CA   (1b)

..3

$\underline{a} \wedge \underline{b} = \underline{c}$;   CC = CA.AND.CB   (1c)

$\underline{a} \vee \underline{b} = \underline{c}$;   CC = CA.OR.CB   (1d)

$\underline{a} \not\equiv \underline{b} = \underline{c}$;   CC = CA.XOR.CB   (1e)

These CL equations are utilized to operate on two-element Boolean vectors of the type $\underline{\underline{a}} = (\underline{a}_{\alpha} \quad \underline{a}_{\beta})$ in SNS. In FORTRAN, these are represented by vectors of the type VA = (AVA, BVA), in which AVA and BVA are Boolean variables of CL, such as those occurring in Eq. (1). In consequence, the two-element vector representing a term, (say $\underline{\underline{a}}$ in SNS), can have four possible states : (1 0), (0 1), (1 1) and (0 0), corresponding to truth ($\underline{\underline{T}}$), falsehood ($\underline{\underline{F}}$), doubt ($\underline{\underline{D}}$) and impossibility ($\underline{\underline{X}}$) (See Refs. [1], [2], [3] for details).

A mathematical machinery was developed whereby the effects of various logical operations acting on the four possible states of VA, (namely STR = (1 0), SFL = (0 1), SDF = (1 1) and SXX = (0 0)), could be worked out. The

..4

details of these, and the essential principles of the computerization of SNS logic were given in Report No. 8[3]. Details of the FORTRAN implementation, in the form of the program NYAYA2, have also been described in Report No. 12[1].

The essential steps of this program have now been implemented on the Texas Instruments programmable pocket calculators (TI 58 and 59). These are briefly described below. This program has been given the name TINY1, the letters TI standing for "Texas Instruments" in the name of the calculator for which it is designed, and the letters NY being the first two letters of the FORTRAN program NYAYA2 which it simulates. The number 1 is used in TINY1, since a second program simulating another FORTRAN program MATLOG for logic is also being planned.

2. ESSENTIALS OF THE LOGICAL APPROACH IN T I N Y 1.

(a) Classical logic operators.

As in the case of the FORTRAN program, the logical

..5

operations of classical logic mentioned above are first implemented. The operators that are programed in TINY1 form a subset of those in the FORTRAN program NYAYA2. These consist firstly, of the classical operators standing for the logical connectives "equals", "not", "or" and "and", whose subroutines have the labels EE, NOP, SUM, PRD respectively (See the Appendix for a listing) where these occur from line 000 to line 038.

A brief account of these four routines is as follows:

(i) Label EE for "equals".

One of inputs is put in _t_-register, and the other is put in _x_-register (display register) as this routine is called. The check x = t is performed. If it is satisfied, the display is made 1, otherwise the display is made 0 and returned.

(ii) Label NOP for "not".

The input is put in the display register as this routine is called. The value in the display is negated with +/- and

..6

1 is added to it and the display value is returned, to give the negation of the input.

(iii) Label SUM for "or".

One of the inputs is put in *t*-register and the other is put in the *x*-register (display register). The *x*-register value and the *t*-register value are added and put in *t*-register. The display is made 0. The check $x < t$ is made. If it is satisfied, the display is made 1; otherwise the display is kept as it is (=0) and returned.

(iv) Label PRD for "and".

One of the inputs is put in the *t*-register and the other is put in the *x*-register (display register). The *x*-register value and the *t*-register value are multiplied and returned.

The extension of these to SNS, consisting of two-element Boolean state vectors, ad operators acting on these, was carried out as follows.

..7

(b) State Vectors.

The four possible states of a vector $\underline{\underline{a}}$, defined as $(\underline{a}_{\alpha} \quad \underline{a}_{\beta})$ are (1 0), (0 1), (1 1) and (0 0). The first three of these, namely $\underline{\underline{T}}$, $\underline{\underline{F}}$ and $\underline{\underline{D}}$, are represented by 1.0, 0.1 and 1.1 respectively, in TINY1. The last one represents the impossible state $\underline{\underline{X}}$, which can never be an original input, but which can occur as the result of a contradiction in the previous step, when it takes the form 1. E -99. If all the inputs are not $\underline{\underline{X}}$, this state can occur only as the output of the SNS connectives "with" and "reverse or" in this program, when it appears as a flashing output. Hence in the routines for "with" and "reverse or", we perform the operation 1/$\underline{x}$ twice on the result, so that 0.0 will be flashed as 1. E -99. However, if we have fixed the number of decimal digits to be displayed as 1, this result will flash as 0.0.

The routine which gives components of a state vector, has

..8

been labelled STO. This routine STO can give the components of two state vectors, (one stored in the t-register ($\underline{\underline{a}}$) and the other in the x-register ($\underline{\underline{b}}$)). A brief account of the program of this is as follows:

(i) The state vector in the t-register ($\underline{\underline{a}}$) is stored in R 00, and the state vector in the display register ($\underline{\underline{b}}$) is stored in R 01 (where R stands for "Register").

(ii) The integer part of the value in the display ($\underline{\underline{b}}$) is obtained ($\underline{b}_\alpha$), and it is stored in R 02. The procedure display $\longrightarrow$ (- (display) + R 01) * 10 gives $\underline{b}_\beta$ and it is stored in R 03.

(iii) The same procedure (ii) is done on $\underline{\underline{a}}$ (originally stored in R 00) and $\underline{a}_\alpha$ is stored in R 00; $\underline{a}_\beta$ in R 01.

Thus this routine STO gives the α and β components of the state vector stored in the t-register, so as to be in R 00 and R 01 and that of the state vector in the display register, to be in R 02 and R 03 respectively. This routine occurs in

..9

the program from line 064 to line 100.

(c) $\underline{\underline{X}}$-priority rule.

Except for the SNS operators upon, with and agree, for all the remaining operators for which routines have been written, namely "not" ($\underset{\sim}{N}$), "implies" ($\underset{\sim}{Y}$), "implicates" ($\underset{\sim}{V} = \overset{\leftarrow}{\underset{\sim}{Y}}$), "and" ($\underset{\sim}{A}$), "or" ($\underset{\sim}{O}$), "binary equivalent" ($\underset{\sim}{S}$) and "reverse or" ($\overset{\leftarrow}{\underset{\sim}{O}}$), the $\underline{\underline{X}}$-priority rule has to be applied (see ref. [2]). This rule requires that when the input, in the case of unary operators, or at least one of the inputs, in the case of binary operators, is $\underline{\underline{X}}$, the output should necessarily be $\underline{\underline{X}}$. However, the mathematical machinery used in NYAYA2 is such that for $\underset{\sim}{N}$, $\underset{\sim}{Y}$ and $\overset{\leftarrow}{\underset{\sim}{Y}}$, the $\underline{\underline{X}}$-priority rule is automatically taken care of. So we have to take care of this only for $\underset{\sim}{A}$, $\underset{\sim}{O}$, $\underset{\sim}{S}$ and $\overset{\leftarrow}{\underset{\sim}{O}}$. The technique that we have used for this is to write a separate routine labelled Σ+, which will merge with SBR STO. The SBR Σ+ will check whether at least one of the state vectors (stored in the $\underline{x}$-register and the $\underline{t}$-register) has the state 0.0 and, if so, will

..10

make both the states 0.0 and pass them on to STO. If not, it will just pass them to SBR STO. The essential idea is that, for $\underset{\sim}{A}$, $\underset{\sim}{0}$, $\underset{\sim}{S}$, and $\overleftarrow{\underset{\sim}{0}}$, if atleast one of the inputs is $\underset{=}{X}$, both are made $\underset{=}{X}$ and SBR STO will give 0.0 for all components ($\underline{a}_{\alpha}$, $\underline{a}_{\beta}$, $\underline{b}_{\alpha}$, $\underline{b}_{\beta}$). Hence, except for $\underset{\sim}{A}$, $\underset{\sim}{0}$, $\underset{\sim}{S}$, and $\overleftarrow{\underset{\sim}{0}}$, SBR STO is called to break the state vectors into its components and for these four alone, SBR Σ+ is called for the purpose. This routine occurs in the program from line 039 to line 063.

In actual practice, a slight modification is needed, since the output from with and reverse or for the state $\underset{=}{X}$ is of the form 1. E -99 and not exactly 0.0, although it will be displayed as 0.0, since we fix the display to show only one decimal place. Hence, in the subroutine Σ+, we check whether 0.01 is greater than at least one of the inputs, and if so make both the inputs exactly equal to 0.0 and pass them on to SBR STO. If not, the inputs are passed as such to SBR STO.

..11

<u>(d) Classical logic operators applied to SNS logic.</u>

The operation of these routines is best explained with reference to the SNS connectives "unary not" $\underset{\sim}{N}$ and "binary or" $\underset{\sim}{O}$

<u>Unary not</u>. The logic used is to convert the input $(\underline{a}_\alpha \quad \underline{a}_\beta)$ into the output $(\underline{a}_\beta \quad \underline{a}_\alpha)$. Hence all that is done is to interchange the components. Now SBR STO stores $\underline{a}_\alpha$ in register 00 and $\underline{a}_\beta$ in register 01. Therefore, the result is obtained as (RCL 01 + RCL 00/10), where RCL $\underline{n}$ means that the contents of the data register $\underline{n}$ is recalled to display.

<u>SNS binary or</u>. For this operator the logic is as follows. The input is $(\underline{a}_\alpha \quad \underline{a}_\beta)$, $(\underline{b}_\alpha \quad \underline{b}_\beta)$, and the output is $(\underline{a}_\alpha \oplus \underline{b}_\alpha \quad \underline{a}_\beta \otimes \underline{b}_\beta)$, where $\oplus$ and $\otimes$ refer to classical "or" and classical "and" respectively. Hence the program proceeds as follows:

(a) SBR Σ+ puts $\underline{a}_\alpha$, $\underline{a}_\beta$, $\underline{b}_\alpha$, $\underline{b}_\beta$ in R 00, R 01, R 02, R 03 respectively.

(b) $\underline{a}_\alpha$, $\underline{b}_\alpha$ are recalled, and SBR SUM is called. The

..12

result is stored in R 00.

(c) $\underline{a}_\beta$, $\underline{b}_\beta$ are recalled, and SBR PRD is called.

(d) Display $\longrightarrow$ (Display x 0.1 + RCL 00) gives the result.

The others such as "unary implies" $\underset{\sim}{Y}$, "unary implicates" $\underset{\sim}{V}$, and "binary equivalent" ($\underset{\sim}{S}$) and "binary and" $\underset{\sim}{A}$ are similarly written. Thus, for $\underset{\sim}{V}$, it is as follows.

Unary implicates ($\underset{\sim}{V}$):

(i) SBR STO puts $\underline{a}_\alpha$ and $\underline{a}_\beta$ in R 00 and R 01.

(ii) The contents of the display register ($\underline{a}_\beta$) is put in the t-register. $\underline{a}_\alpha$ is recalled and SBR SUM is called.

(iii) (Display $\longrightarrow$ display x 0.1 + RCL 00) gives the result.

Binary Equivalent ($\underset{\sim}{S}$): This routine is labelled as = . The procedure used for this is different from that used in NYAYA2, but very similar to that used in MATLOG. It is defined as follows:

..13

$$\underline{\underline{a}}\ S\ \underline{\underline{b}} = \underline{\underline{c}} \longmapsto \begin{array}{l} \underline{c}_\alpha = \underline{a}_\alpha \underline{b}_\alpha \oplus \underline{a}_\beta \underline{b}_\beta \ ; \\ \underline{c}_\beta = \underline{a}_\beta \underline{b}_\alpha + \underline{a}_\alpha \underline{b}_\beta \end{array}$$

Briefly, the steps in the program are as follows :

(i) SBR $\Sigma+$ puts $\underline{a}_\alpha$, $\underline{a}_\beta$, $\underline{b}_\alpha$, $\underline{b}_\beta$ in R 00, R 01, R 02, R 03 respectively.

(ii) $\underline{a}_\alpha$ and $\underline{b}_\alpha$ are recalled and the product ($\underline{a}_\alpha \times \underline{b}_\alpha$) found with the result put in the $\underline{t}$-register. $\underline{a}_\beta$ and $\underline{b}_\beta$ are similarly recalled and the product ($\underline{a}_\beta \times \underline{b}_\beta$) found. SBR SUM is called, which gives $\underline{c}_\alpha = \underline{a}_\alpha \underline{b}_\alpha \oplus \underline{a}_\beta \underline{b}_\beta$, and it is put in R 04.

(iii) A similar procedure gives $\underline{c}_\beta$ in the display register.

(iv) (Display $\longrightarrow$ Display $\times$ 0.1 + RCL 04) gives the result, namely the state of $\underline{c}$.

In Table 1, the names of the SNS operators, along with their logical one-letter symbols that we use and the labels of the subroutines are listed, for those operations that are

..14

contained in this program.

The program is in two parts — the Core, and the Subroutines. The core contains the classical logical operators $\vee$, $\wedge$, $\neg$ and $\equiv$. It also contains a routine which will make both the input vectors as $\underline{\underline{X}}$ if even one of them is $\underline{\underline{X}}$ and which will be used by the routines which need the $\underline{\underline{X}}$-priority rule to be applied, and a routine which breaks up a state vector into its two component parts. These are all used by the subroutines, but not called by the user program.

The subroutines part consists of those for the SNS analogs for the classical connectives "and", "or", "equivalent", "not", "implies" and "implicates" and for the pure SNS connectives "with", "upon", "agree" and "reverse or". These have the labels indicated in Table 1. The time required for each

Table 1. Outline of TINY1 with names of Labels and Locations of Subroutines

..15

Table 1

Outline of TINY1 with names of Labels and Locations of Subroutines

Program Location Nos.	Nature	Label
	C O R E	
000–014	Classical OR	SUM
015–020	Classical AND	PRD
021–027	Classical NOT	NOP
028–038	Classical EQ	EE
039–063	$\underline{\underline{X}}$-priority rule	$\Sigma+$
064–100	Components of state vector	STO
	SUBROUTINES	
	SNS analogs of Classical Operators	
101–115	Unary not ($\underset{\sim}{N}$)	–
116–135	Unary implies ($\underset{\sim}{Y}$)	1/X

Contd..

Table 1 Continued.

Program Location Nos.	Nature	Label
136 153	Unary implicates ($\underset{\sim}{V}$)	Y^X
154 197	Binary equivalent ($\underset{\sim}{S}$)	=
198 225	Binary or ($\underset{\sim}{O}$)	+
226 253	Binary and ($\underset{\sim}{A}$)	×
	Purely SNS Operators	
254 283	With ($\underset{\sim}{W}$)	CLR
284 311	Upon ($\underset{\sim}{U}$)	÷
312 348	Agree ($\underset{\sim}{G}$)	$\sqrt{X}$
349 383	Reverse or ($\bar{O}$)	INV

..17

of these routines are about two seconds per call. A short description of the SNS routines are given in the next subsection

<u>(e) Purely SNS operators</u>.

There are three purely SNS operators $\underset{\sim}{W}$, $\underset{\sim}{U}$ and $\underset{\sim}{G}$. Since they require logical equations different from classical logic, they were programmed by using the matrix representation, and defining these as follows:

$$\underline{\underline{a}}\,\underset{\sim}{W}\,\underline{\underline{b}} = \underline{\underline{c}} \longmapsto \underline{c}_\alpha = \underline{a}_\alpha \otimes \underline{b}_\alpha \;;\; \underline{c}_\beta = \underline{a}_\beta \otimes \underline{b}_\beta \qquad (2a)$$

$$\underline{\underline{a}}\,\underset{\sim}{U}\,\underline{\underline{b}} = \underline{\underline{c}} \longmapsto \underline{c}_\alpha = \underline{a}_\alpha \oplus \underline{b}_\alpha \;;\; \underline{c}_\beta = \underline{a}_\beta \oplus \underline{b}_\beta \qquad (2b)$$

$$\begin{aligned}\underline{\underline{a}}\,\underset{\sim}{G}\,\underline{\underline{b}} = \underline{\underline{c}} \longmapsto\ & \underline{\underline{c}} = \underline{\underline{T}} \quad \text{if } \underline{a}_\alpha = \underline{b}_\beta \text{ and } \underline{a}_\beta = \underline{b}_\beta\,, \\ & \text{and } \underline{\underline{c}} = \underline{\underline{F}} \text{ otherwise}\end{aligned} \qquad (2c)$$

The last one in (2c) is implemented as follows:

$$\begin{aligned}\underline{\underline{a}}\,\underset{\sim}{G}\,\underline{\underline{b}} = \underline{\underline{c}} \longmapsto\ & \underline{c}_\alpha = (\underline{a}_\alpha \equiv \underline{b}_\alpha) \otimes (\underline{a}_\beta \equiv \underline{b}_\beta); \\ & \underline{c}_\beta = \neg\,\underline{c}_\alpha\end{aligned} \qquad (2d)$$

Since for $\underset{\sim}{W}$, $\underset{\sim}{U}$ and $\underset{\sim}{G}$, the $\underline{\underline{X}}$-priority rule need not be applied, SUB STO is used to get the components in the rout~~s~~ines for all these three.

--18

With: For $\underset{\sim}{W}$, the program proceeds as follows:

(i) SBR STO puts $\underline{a}_{\alpha}$, $\underline{a}_{\beta}$, $\underline{b}_{\alpha}$, $\underline{b}_{\beta}$ in R 00, R 01, R 02, R 03 respectively.

(ii) $\underline{a}_{\alpha}$, $\underline{b}_{\alpha}$ are recalled, and SBR PRD is called. The result is stored in R00.

(iii) $\underline{a}_{\beta}$, $\underline{b}_{\beta}$ are recalled, and SBR PRD is called.

(iv) Display $\longrightarrow$ (Display $\times$ 0.1 + RCL 00) gives the result.

(v) Since $\underset{\sim}{W}$ can generate an $\underline{\underline{X}}$ output, finally the steps 1/$\underline{x}$, 1/$\underline{x}$ is added. For non $\underline{\underline{X}}$ output 1/$\underline{x}$, 1/$\underline{x}$ gives the same value. For 0.0 alone, it gives out a flashing display value 1. E -99 indicating that a contradiction has arisen.

Upon: For $\underset{\sim}{U}$, the program proceeds as follows:

(i) SBR STO puts $\underline{a}_{\alpha}$, $\underline{a}_{\beta}$, $\underline{b}_{\alpha}$, $\underline{b}_{\beta}$ in R 00, R 01, R 02, R 03 respectively.

(ii) $\underline{a}_{\beta}$, $\underline{b}_{\beta}$ are recalled and SBR SUM is called. The result is stored in R 01.

..19

(iii) $\underline{a}_\alpha$, $\underline{b}_\alpha$ are recalled and SBR SUM is called.

(iv) Display $\longrightarrow$ (Display + RCL 01 * 0.1) gives the result.

Agree:For $\underset{\sim}{G}$, the program proceeds as follows :

(i) SBR STO puts $\underline{a}_\alpha$, $\underline{a}_\beta$, $\underline{b}_\alpha$, $\underline{b}_\beta$ in R 00, R 01, R 02, R 03

respectively.

(ii) $\underline{a}_\alpha$, $\underline{b}_\alpha$ are recalled and SBR EE is called and the result

is stored in R 00.

(iii) $\underline{a}_\beta$, $\underline{b}_\beta$ are recalled and SBR EE is called and the result

is put in the $\underline{t}$-register

(iv) R 00 is recalled and SBR PRD is called and the result

is stored in R 00.

(v) SBR NOP is called, and

(Display $\longrightarrow$ Display $\times$ 0.1 + RCL 00) gives the result.

<u>(f) Reverse operators</u>.

As far as reverse operators are concerned, routines have been written only for "reverse or" $(\overleftarrow{\underset{\sim}{O}})$ and "reverse if" $(\overleftarrow{\underset{\sim}{Y}} \equiv \underset{\sim}{V})$.

..20

For "reverse and" ($\overset{\leftarrow}{\underset{\sim}{A}}$), we have not written routines since it can be expressed in terms of $\overset{\leftarrow}{\underset{\sim}{O}}$ and $\underset{\sim}{N}$ as shown in the Eq. (3) below :

$$(\underline{\underline{c}}\ \overset{\leftarrow}{\underset{\sim}{A}}\ \underline{\underline{a}} = \underline{\underline{b}}) \equiv ((\underline{\underline{c}}\ \underset{\sim}{N})\ \overset{\leftarrow}{\underset{\sim}{O}}\ (\underline{\underline{a}}\ \underset{\sim}{N}) = \underline{\underline{b}}\ \underset{\sim}{N}) \qquad (3)$$

The routine for the Unary implicates $\underset{\sim}{V}$ ($\equiv \overset{\leftarrow}{\underset{\sim}{Y}}$) has already been described. The essential algebra that is employed to implement $\overset{\leftarrow}{O}$ is as follows.

$$\underline{\underline{c}}\ \overset{\leftarrow}{\underset{\sim}{O}}\ \underline{\underline{a}} = \underline{\underline{b}} \longmapsto \underline{b}_{\alpha} = \underline{c}_{\alpha} \quad ;$$

$$\underline{b}_{\beta} = (\underline{c}_{\alpha} \otimes \underline{a}_{\alpha}) \oplus (\underline{c}_{\beta} \otimes \underline{a}_{\beta}) \qquad (4)$$

<u>Reverse or</u>.

Here the routine that is called to get the components of input vectors is SBR $\Sigma+$, since it does not automatically lead to the consequences of the $\underline{\underline{X}}$-priority rule. The steps are as follows.

..21

(i) SBR Σ+ puts $\underline{c}_{\alpha}$, $\underline{c}_{\beta}$, $\underline{a}_{\alpha}$ and $\underline{a}_{\beta}$ in the data registers R 00, R 01, R 02 and R 03 respectively.

(ii) $\underline{c}_{\alpha}$ and $\underline{a}_{\alpha}$ are recalled and the SBR PRD is called. The result is stored in R 02.

(iii) $\underline{c}_{\beta}$ and $\underline{a}_{\beta}$ are recalled, the SBR PRD is called, and the result is put in the $\underline{t}$-register. The contents of R 02 is recalled and SBR SUM is called.

(iv) (Display $\longrightarrow$ Display x 0.1 + RCL 00) gives the result.

(v) To obtain a flashing output in case $\underline{\underline{X}}$ is obtained, 1/$\underline{x}$ is performed twice.

3. METHOD OF OPERATION

In relation to the FORTRAN program, only a small number of operators are defined in TINY1, although they are more than the absolute minimum that is needed in classical logic — namely $\underset{\sim}{N}$ and $\underset{\sim}{O}$, or $\underset{\sim}{N}$ and $\underset{\sim}{A}$. We have defined $\underset{\sim}{N}$, $\underset{\sim}{Y}$, $\underset{\sim}{S}$, $\underset{\sim}{O}$ and $\underset{\sim}{A}$, as these are very commonly required. The other logical connec-

..22

tives such as "nand" and "nor", can be obtained as follows:

$$\underline{\underline{a}}\ \text{"nand"}\ \underline{\underline{b}} = (\underline{\underline{a}}\ \underset{\sim}{N})\ \underset{\sim}{O}\ (\underline{\underline{b}}\ \underset{\sim}{N}) \qquad (5a)$$

$$\underline{\underline{a}}\ \text{"nor"}\ \underline{\underline{b}} = (\underline{\underline{a}}\ \underset{\sim}{N})\ \underset{\sim}{A}\ (\underline{\underline{b}}\ \underset{\sim}{N}) \qquad (5b)$$

Since the connectives for "if" and "only if" are frequently used, the converse of $\underset{\sim}{Y}$ (implies), namely $\underset{\sim}{V}$ (implicates), has also been implemented. All these operators can be applied one after the other without any difficulty, as they essentially have the transitive property. The purpose and use of $\underset{\sim}{W}$, $\underset{\sim}{U}$ and $\underset{\sim}{G}$ have been described in [1] on FORTRAN programing, and in [2] and [3]. Therefore, no details are mentioned here.

A summary of the user instructions for this program is shown in Table 2, and the Appendix provides a complete print out, obtained using the printer associated with TI 58/59, and it contains in it information as to which parts of it correspond to the logic that has been programmed by it.

Table 2. User Instructions for Subroutines in TINY1

..23

Table 2. User Instructions for Subroutines in TINY1

Operator name	Effective logical operation	Instruction	t-register	Display
$\underset{\sim}{N}$	not $\underline{\underline{a}}$	SBR –	–	$\underline{\underline{a}}$
$\underset{\sim}{Y}$	$\underline{\underline{a}}$ implies	SBR 1/X	–	$\underline{\underline{a}}$
$\underset{\sim}{V}$ ($\underset{\sim}{\overleftarrow{Y}}$)	$\underline{\underline{a}}$ implicates	SBR Y^X	–	$\underline{\underline{a}}$
$\underset{\sim}{S}$	$\underline{\underline{a}} \equiv \underline{\underline{b}}$	SBR =	$\underline{\underline{a}}$	$\underline{\underline{b}}$
$\underset{\sim}{A}$	$\underline{\underline{a}}$ and $\underline{\underline{b}}$	SBR ×	$\underline{\underline{a}}$	$\underline{\underline{b}}$
$\underset{\sim}{O}$	$\underline{\underline{a}}$ or $\underline{\underline{b}}$	SBR +	$\underline{\underline{a}}$	$\underline{\underline{b}}$
$\underset{\sim}{W}$	$\underline{\underline{a}}$ with $\underline{\underline{b}}$	SBR CLR	$\underline{\underline{a}}$	$\underline{\underline{b}}$
$\underset{\sim}{U}$	$\underline{\underline{a}}$ upon $\underline{\underline{b}}$	SBR ÷	$\underline{\underline{a}}$	$\underline{\underline{b}}$
$\underset{\sim}{G}$	$\underline{\underline{a}}$ agree $\underline{\underline{b}}$	SBR $\sqrt{X}$	$\underline{\underline{a}}$	$\underline{\underline{b}}$
$\underset{\sim}{\overleftarrow{O}}$	If $\underline{\underline{a}}\ \overleftarrow{O}\ \underline{\underline{b}} = \underline{\underline{c}}$, what is $\underline{\underline{b}}$, given $\underline{\underline{c}}$ and $\underline{\underline{a}}$?	SBR INV	$\underline{\underline{c}}$	$\underline{\underline{a}}$

Instructions:

For unary operations, enter $\underline{\underline{a}}$, press the instruction, namely SBR & then the relevant key for the label.

For binary operations, enter $\underline{\underline{a}}$, press x $\rightleftarrows$ t, enter $\underline{\underline{b}}$, press SBR & then the relevant key for the label.

(Note: The order of entering two terms is important only for $\overleftarrow{O}$, for which $\underline{\underline{c}}$ has to be entered first, and then $\underline{\underline{a}}$).

..24

The full program takes 384 places in the memory, and there are 100 more places available for putting in the running program for an actual problem, even in TI 58 with 480 memory locations. As will be seen from the examples given in Section 4, the program for a medium-sized problem takes about 75 locations. Hence, even the smaller TI 58 can be used, provided only the subroutines required for the particular problem are put in, along with the core of the program. Anticipating what will be discussed in Section 4(b), the detective problem, worked out in a straightforward manner, requires only the following subroutines: $\underset{\sim}{N}$ (SBR −), $\underset{\sim}{S}$ (SBR =), $\underset{\sim}{A}$ (SBR ×), $\underset{\sim}{O}$ (SBR +), $\underset{\sim}{U}$ (SBR ÷) and $\underset{\sim}{G}$ (SBR $\sqrt{X}$), which, together with the core, take up 281 locations in addition to another 86 locations for the program (Label B) of the problem itself. Four data registers are needed for the main program and four more for the problem. But we can set the data registers in sets of 10. Thus, the total number of locations needed is 367 + 10 × 8 = 447, so that the memory

..25

space required for this problem is available within the range of TI 58 itself. On the other hand, TI 59 has 960 memory locations, and hence the full program listed in the Appendix, along with the programs of all the three problems considered in the next section, under Labels A, B and C, can be contained in it, with still more space to spare. Hence, the TI 59, with facility for using magnetic cards, is ideal for using the program discussed here for applications and for demonstrations.

4. EXAMPLES OF PRACTICAL APPLICATIONS

The steps involved in using TINY1 for particular problems can be illustrated by the following three examples — namely, the "Party and Queue" problem of Chapter 10 of Ref. [2], and the detective problem, as shown in Fig. 1 and Fig. 2 of Report No. 12 [1]. In each case, the actual running program that was fed is also shown with samples of results.

.. 26

a) Party and Queue Problem.

The details of the problem may be obtained from Ref. [2] and we give here only the logical equations, the corresponding logical graph, the program to be fed to TINY1 for implementation on the calculator, and the results obtained. The logical equations are as follows:

$$\underline{\underline{a}}\ \overset{\leftarrow}{\underset{\sim}{A}}\ \underline{\underline{b}} = \underline{\underline{g}} \qquad (6a)$$

$$\underline{\underline{c}}\ \overset{\leftarrow}{\underset{\sim}{O}}\ \underline{\underline{g}} = \underline{\underline{x}} \qquad (6b)$$

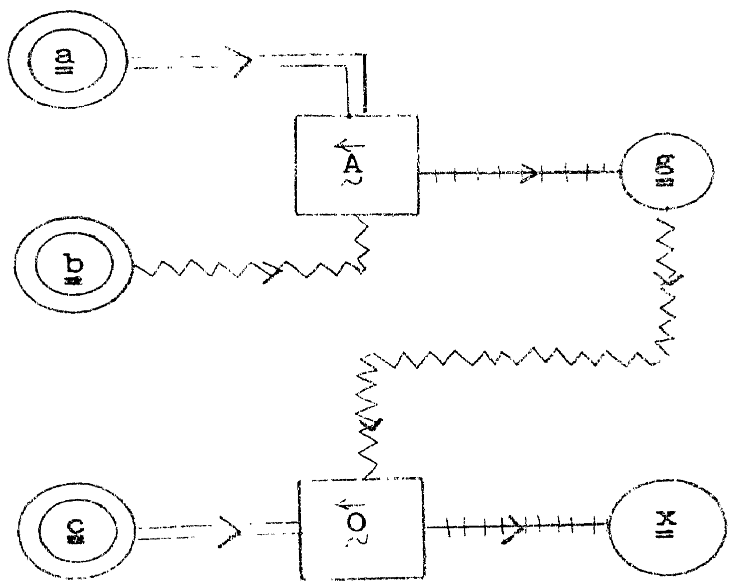
The logical graph is as shown in Fig. 1(a) and the notation and symbology employed in the graph are the same as those described in Ref. [2].

However, since, we have not written the routine for $\overset{\leftarrow}{\underset{\sim}{A}}$ in TINY1 (for reasons already mentioned), we shall modify the logical graph by expressing $\overset{\leftarrow}{\underset{\sim}{A}}$ in terms of $\overset{\leftarrow}{\underset{\sim}{O}}$ as

$$\left[\underline{\underline{a}}\ \overset{\leftarrow}{\underset{\sim}{A}}\ \underline{\underline{b}} = \underline{\underline{g}}\right] \equiv \left[((\underline{\underline{a}}\ \underset{\sim}{N})\ \overset{\leftarrow}{\underset{\sim}{O}}\ (\underline{\underline{b}}\ \underset{\sim}{N}))\underset{\sim}{N} = \underline{\underline{g}}\right] \qquad (6c)$$

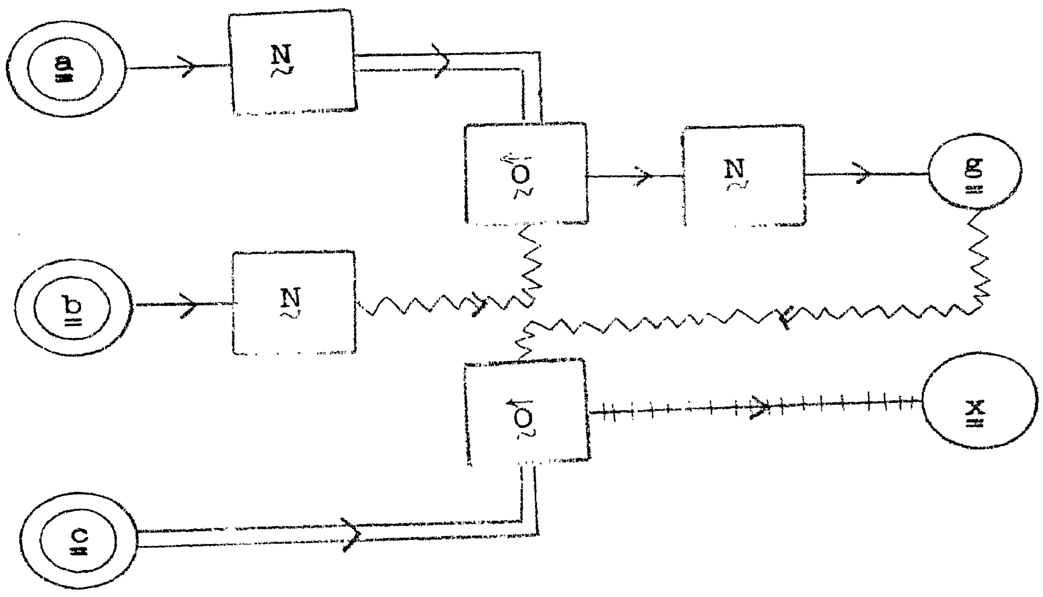
Fig. 1(a) Party and queue problem, as in Eq. (6)
(b) The same problem suitably modified for use with TINY1.

..27



$\underline{\underline{a}}$
$\overleftarrow{\underset{\sim}{A}}$
$\underline{\underline{g}}$
$\underline{\underline{b}}$
$\underline{\underline{c}}$
$\overleftarrow{\underset{\sim}{O}}$
$\underline{\underline{x}}$

Fig. 1(a) Party and Queue problem as in Eq. 6



$\underline{\underline{a}}$
$\underset{\sim}{N}$
$\overleftarrow{\underset{\sim}{O}}$
$\underset{\sim}{N}$
$\underline{\underline{g}}$
$\underline{\underline{b}}$
$\underset{\sim}{N}$
$\overleftarrow{\underset{\sim}{O}}$
$\underline{\underline{x}}$
$\underline{\underline{c}}$

Fig. 1(b) The same problem suitably modi[illegible]

The new, but equivalent, logical graph of the problem is shown in Fig. 1(b). The program used for the problem (lines 384 to 425, labelled A) is listed in Table 3(a). The steps are obvious from the graph shown above. Table 3(b) shows the print out obtained for 8 sets of inputs for a, b and c. These inputs are the first three in each printed set of 5 terms, the fourth and fifth being g and x are given below in Table 3. For ready reference, the results have been retyped in tabular form in Table 3c.

Table 3(a) Program used for the Party and Queue problem.
(b) Printout from the calculator.
(c) The results retyped for convenient reference.

b) Detective problem given in Ref. [1] — Normal Procedure.

Since the logical graph and the details about this problem are given in Ref. [1], we shall give here only the logical equations, corresponding to Fig. 2 below:

..29

Table 3(a). Program used for the party and queue problem

[illegible]	[illegible]	LBL
[illegible]		5
[illegible]	[illegible]	STO
[illegible]	[illegible]	10
[illegible]	[illegible]	PRT
[illegible]		[illegible]
[illegible]	[illegible]	STO
		[illegible]
[illegible]	[illegible]	PRT
[illegible]		R/S
[illegible]	[illegible]	[illegible]
[illegible]		[illegible]
[illegible]		PRT
[illegible]	[illegible]	[illegible]L
[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	-
[illegible]	[illegible]	STO
[illegible]	[illegible]	13
[illegible]	[illegible]	RCL
[illegible]		11
[illegible]	[illegible]	SBR
[illegible]	[illegible]	-
[illegible]	[illegible]	X↕T
[illegible]	[illegible]	RCL
[illegible]	[illegible]	13
[illegible]	[illegible]	X↕T
[illegible]	[illegible]	SBR
[illegible]	[illegible]	INV
[illegible]	[illegible]	SBR
[illegible]	[illegible]	-
415	[illegible]	PRT
416	[illegible]	R/S
[illegible]	3[illegible]	[illegible]T
[illegible]	43	RCL
[illegible]	[illegible]	12
[illegible]	[illegible]	X↕T
42[illegible]	[illegible]	SBR
[illegible]	[illegible]	INV
[illegible]	[illegible]	PRT
[illegible]	[illegible]	ADV
[illegible]	[illegible]	R/S

Table 3(b). Printout from the calculator

```
[illegible]
```

Table 3(c) Results of the Partv and Queue problem.

$\underline{\underline{a}}$	$\underline{\underline{b}}$	$\underline{\underline{c}}$	$\underline{\underline{g}}$	$\underline{\underline{x}}$
1.0	1.0	1.0	1.0	1.1
1.0	1.0	0.1	1.0	0.0
1.0	0.1	1.0	0.0	0.0
1.0	0.1	0.1	0.0	0.0
0.1	1.0	1.0	0.1	1.0
0.1	1.0	0.1	0.1	0.1
0.1	0.1	1.0	1.1	1.1
0.1	0.1	0.1	1.1	0.1

..31

$$\underline{\underline{p}}\ \underset{\sim}{O}\ \underline{\underline{q}} = \underline{\underline{a}} \qquad (7a)$$

$$(\underline{\underline{r}}\ \underset{\sim}{N})\ \underset{\sim}{A}\ \underline{\underline{p}} = \underline{\underline{b}} \qquad (7b)$$

$$\underline{\underline{q}}\ \underset{\sim}{\not{S}}\ \underline{\underline{r}} = \underline{\underline{c}} \qquad (7c)$$

$$\underline{\underline{T}}\ \underset{\sim}{G}\ \underline{\underline{a}} = \underline{\underline{u1}} \qquad (7d)$$

$$\underline{\underline{T}}\ \underset{\sim}{G}\ \underline{\underline{b}} = \underline{\underline{u2}} \qquad (7e)$$

$$\underline{\underline{T}}\ \underset{\sim}{G}\ \underline{\underline{c}} = \underline{\underline{u3}} \qquad (7f)$$

$$\underline{\underline{u1}}\ \underset{\sim}{U}\ \underline{\underline{u2}}\ \underset{\sim}{U}\ \underline{\underline{u3}} = \underline{\underline{x}} \qquad (7g)$$

Fig. 2 Logical graph of the Detective problem (Normal procedure), as adapted for TINY1.

These equations are programmed as such except for the third equation. Since the FORTRAN operator $\underset{\sim}{\not{S}}$ is not available, we use the equivalence

$$\left[\underline{\underline{q}}\ \underset{\sim}{\not{S}}\ \underline{\underline{r}} = \underline{\underline{c}}\right] \equiv \left[(\underline{\underline{q}}\ \underset{\sim}{S}\ \underline{\underline{r}})\underset{\sim}{N} = \underline{\underline{c}}\right] \qquad (7h)$$

Fig. 2 contains this modification, and corresponds exactly to the program fed in. The listing of program under Label B (lines 426 to 512) is given in Table 4(a), and the results obtained using the printer are shown in Table 4(b). Here,

..32

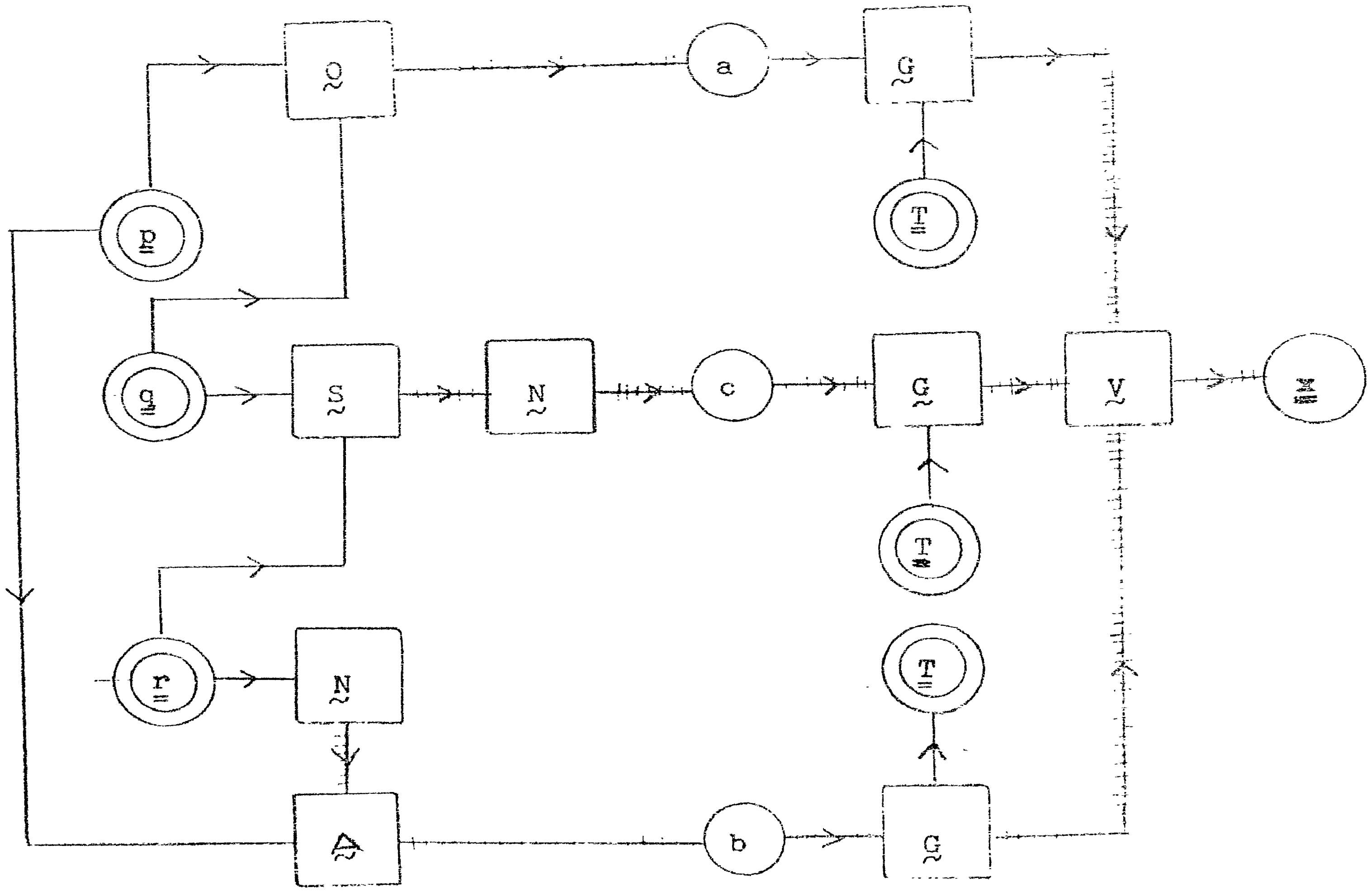


Fig.2. Logical graph of the Detective problem (Normal procedure), as adapted for TINY 1.

the data printed out have been cut and repasted. The values of the inputs $\underline{\underline{p}}$, $\underline{\underline{q}}$, $\underline{\underline{r}}$ and the outputs $\underline{\underline{a}}$, $\underline{\underline{b}}$, $\underline{\underline{c}}$ and $\underline{\underline{x}}$ are presented for the eight possible combination of truth values $\underline{\underline{T}}$ (1.0) and $\underline{\underline{F}}$ (0.1) for each of $\underline{\underline{p}}$, $\underline{\underline{q}}$ and $\underline{\underline{r}}$.

Table 4(a) Printout of the program for the Detective problem (Normal procedure).
(b) Output data from the printer, rearranged for ready reference.

---

c) <u>Detective problem</u> — <u>shortened procedure</u>.

The logical equations are given below:

$$\underline{\underline{T}}\ \overleftarrow{\underset{\sim}{A}}\ (\underline{\underline{r}}\ \underset{\sim}{N}) = \underline{\underline{p1}} \qquad (8a)$$

$$\underline{\underline{T}}\ \overleftarrow{\underset{\sim}{S}}\ \underline{\underline{r}} = \underline{\underline{q}} \qquad (8b)$$

$$\underline{\underline{T}}\ \overleftarrow{\underset{\sim}{O}}\ \underline{\underline{q}} = \underline{\underline{p2}} \qquad (8c)$$

$$\underline{\underline{p1}}\ \underset{\sim}{W}\ \underline{\underline{p2}} = \underline{\underline{p}} \qquad (8d)$$

Since no routine is available for $\overleftarrow{\underset{\sim}{A}}$, the first equation is reqritten using $\overleftarrow{\underset{\sim}{O}}$, as follows:

$$(\underline{\underline{F}}\ \overleftarrow{\underset{\sim}{O}}\ \underline{\underline{r}})\underset{\sim}{N} = \underline{\underline{p1}} \qquad (8e)$$

..34

Table 4(a) Printout of the program for the Detective problem (Normal procedure)

```
426  [illegible] LBL
427  [illegible] B
428  [illegible] STO
429  [illegible] 10
430  [illegible] PRT
431  [illegible] R/S
432  [illegible] STO
433  [illegible] 11
434  [illegible] PRT
435  [illegible] R/S
436  [illegible] STO
437  [illegible] 12
438  [illegible] PRT
439  [illegible] RCL
440  [illegible] 10
441  [illegible] X:T
442  [illegible] RCL
443  [illegible] 11
444  [illegible] SBR
445  [illegible] [illegible]
446  [illegible] STO
447  [illegible] 13
448  [illegible] PRT
449  [illegible] R/S
450  [illegible] RCL
451  [illegible] 12
452  [illegible] SBR
453  [illegible] -
454  [illegible] X:T
455  [illegible] RCL
456  10  10
457  [illegible] SBR
458  [illegible] X
459  [illegible] STO
460  14  14
461  [illegible] PRT
462  [illegible] R/S
463  [illegible] RCL
464  [illegible] 11
465  [illegible] X:T
466  [illegible] RCL
467  [illegible] 12
468  [illegible] SBR
469  [illegible] =
470  71  SBR
471  75  -
472  99  PRT
473  91  R/S
474  [illegible] X:T
475  01  1
476  93  .
477  00  0
478  71  SBR
479  34  ΓX
480  42  STO
481  15  15
482  01  1
483  93  .
484  00  0
485  [illegible] [illegible]
486  43  RCL
487  13  13
488  71  SBR
489  34  ΓX
490  42  STO
491  16  16
492  01  1
493  93  .
494  00  0
495  32  X:T
496  43  RCL
497  14  14
498  71  SBR
499  34  ΓX
500  32  X:T
501  43  RCL
502  15  15
503  71  SBR
504  55  ÷
505  32  X:T
506  43  RCL
507  16  16
508  71  SBR
509  55  ÷
510  99  PRT
511  98  ADV
512  91  R/S
```

Table 4(b) Output data from the printer, rearranged for ready reference

p	0	0	[illegible]	0	0. 1	0.	1.	[illegible]
q	[illegible]	[illegible]	0. 1	0. 1	1. 0	1. 0	0. 1	[illegible]
r	[illegible]	0. 1	1. 0	0. 1	1. 0	0. 1	1. 0	[illegible]
a	[illegible]	[illegible]	1. 0	1. 0	1. 0	1. 0	0. 1	[illegible]
b	0.	0	0. 1	1. 0	0. 1	0. 1	0. 1	[illegible]
c	1	[illegible]	1. 0	0. 1	0. 1	1. 0	1. 0	[illegible]
x	[illegible]	1. 0	1. 1	1. 1	1. 1	1. 1	1. 1	[illegible]

similarly, (8b) is capable of being converted to the simple form

$$\underline{\underline{r}}\,\underset{\sim}{N} = \underline{\underline{q}} \qquad (8f)$$

The logical graph, as implemented in TINY1, is shown in Fig. 3.

Fig. 3. Logical graph of the Detective problem (Shortened procedure).

The listing of the program, with the changes, as in (8e) and (8f), and the results as obtained from the printer are shown in Table 5(a) and (b). As in the problem (b), here also, the output has been cut and repasted for ready reference.

Table 5(a) Printout of the program for the Detective problem (Shortened procedure).
(b) Output data from the printer, rearranged for ready reference.

..36

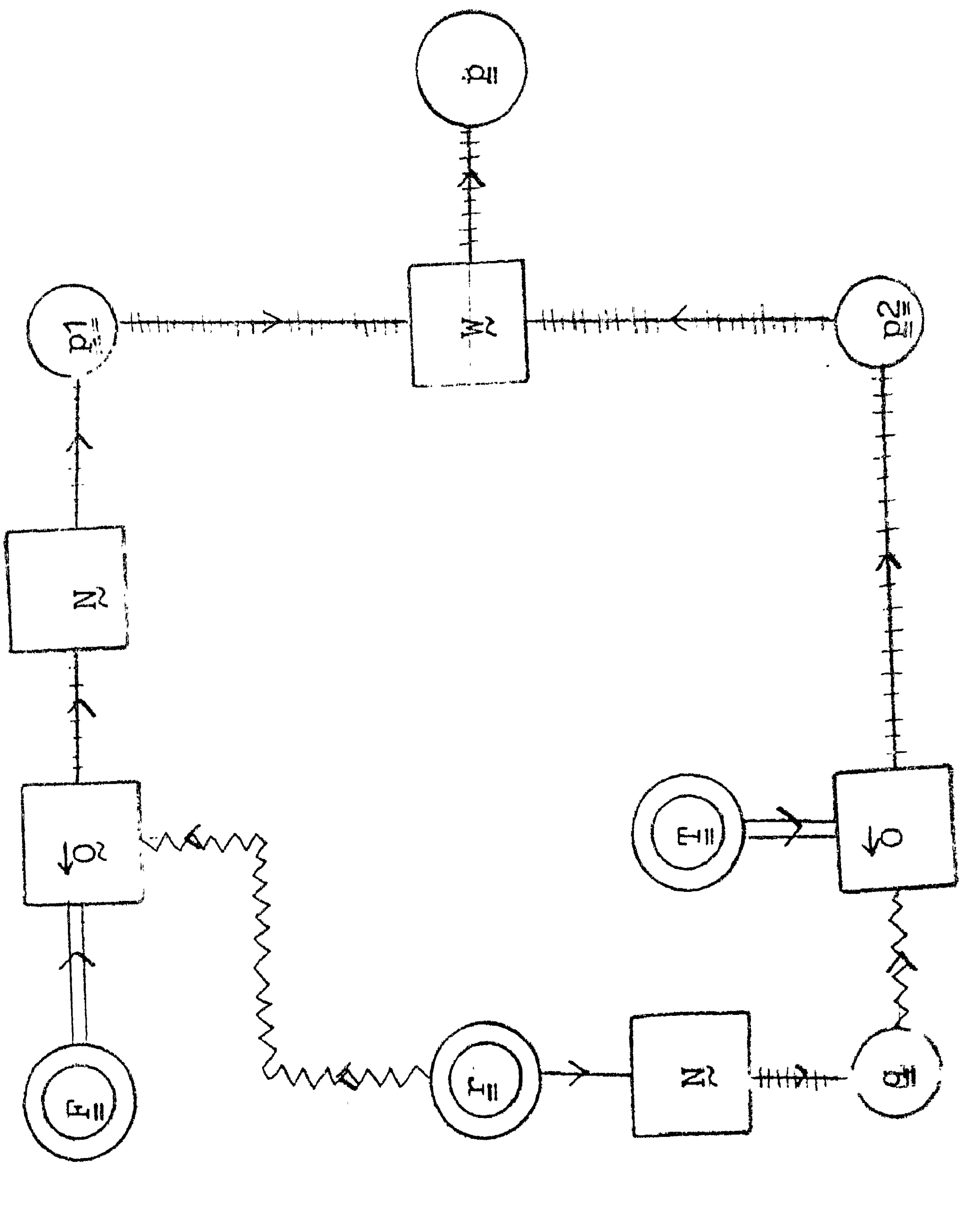


Fig.3. Logical graph of the Detective problem (shortened procedure)

Table 5(a). Printout of the program for the Detective problem (Shortened procedure)

```
513   76  LBL
514   10   E'
515   42  STO
516   10   10
517   99  PRT
518   00    0
519   93    .
520   01    1
521   32  X:T
522   43  RCL
523   10   10
524   71  SBR
525   22  INV
526   71  SBR
527   75    -
528   99  PRT
529   91  R/S
530   42  STO
531   11   11
532   43  RCL
533   10   10
534   71  SBR
535   75    -
536   99  PRT
537   91  R/S
538   32  X:T
539   01    1
540   93    .
541   00    0
542   32  X:T
543   71  SBR
544   22  INV
545   99  PRT
546   91  R/S
547   32  X:T
548   43  RCL
549   11   11
550   71  SBR
551   25  CLR
552   99  PRT
553   91  R/S
```

Table 5(b). Output data from the printer, rearranged for ready reference

r	1.0		0.1
p 1	0.0	?	[illegible]
q	0.1		1.0
p 2	1.0		1.
p	0.0	?	1.0

5. Concluding Remarks

Thus it will be seen that the program TINY1 can be used to work out logical programs of reasonable magnitude. The program has been designed in such a way as to use the minimum number of subroutines for the various logical connectives, subject to reasonable convenience. It has been tested for various problems and found to be convenient to operate.

As mentioned earlier, this program TINY1 is an adaptation of the FORTRAN program NYAYA2 which makes use of logical operations using one element Boolean variables, which are available in FORTRAN-10. After this was written, we have developed another approach to Sentential Logic purely in terms of Boolean vectors and matrices and without reference to logical operations *per se* available in FORTRAN. It has been tested and we hope to develop a version of it for TI59 in due course.

REFERENCES

1. Ramachandran, G.N., and Thanaraj, T.A, "FORTRAN PROGRAM FOR SENTENTIAL LOGIC" - Matphil Reports No. 12, 1980.

2. Ramachandran, G.N. "Principles of Practical Logic", (under preparation)

3. Ramachandran, G.N. "Computerization of Logic", Golden Jubilee Commemoration Volume, Natl. Acad. Sci. India, Allahabad. 1980, pp 1-42. Matphil Reports No. 8.

4. Ramachandran, G.N, and Thanaraj, T.A. "FORTRAN program MATLOG for sentential logic". (under preparation)

APPENDIX : LISTING OF TINY 1 PROGRAM

Core of the Program

```
Classical OR
[illegible]  [illegible]  LBL
[illegible]  [illegible]  [illegible]
[illegible]  [illegible]  -
[illegible]  [illegible]  X:T
[illegible]  [illegible]  =
[illegible]  [illegible]  [illegible]
[illegible]  [illegible]  0
[illegible]  [illegible]  [illegible]
[illegible]  [illegible]  GE
[illegible]  [illegible]  [illegible]
[illegible]  [illegible]  [illegible]
[illegible]  [illegible]  RTN
[illegible]  [illegible]  [illegible]
[illegible]  [illegible]  =
[illegible]  [illegible]  RTN

Classical AND
[illegible]  [illegible]  LBL
[illegible]  [illegible]  RCL
[illegible]  [illegible]  [illegible]
[illegible]  [illegible]  X:T
[illegible]  [illegible]  =
[illegible]  [illegible]  RTN

Classical NOT
[illegible]  [illegible]  LBL
[illegible]  [illegible]  [illegible]
[illegible]  [illegible]  -
[illegible]  [illegible]  [illegible]
[illegible]  [illegible]  1
[illegible]  [illegible]  =
[illegible]  [illegible]  RTN

Classical IF
[illegible]  [illegible]  LBL
[illegible]  [illegible]  GE
[illegible]  [illegible]  [illegible]
[illegible]  [illegible]  00
[illegible]  [illegible]  36
[illegible]  [illegible]  0
[illegible]  [illegible]  =
035  [illegible]  RTN
036  [illegible]  1
037  [illegible]  =
038  [illegible]  RTN

X-priority
039  [illegible]  LBL
040  [illegible]  [illegible]
041  [illegible]  STO
042  [illegible]  00
043  [illegible]  [illegible]
044  [illegible]  0
045  [illegible]  1
046  [illegible]  GE
047  [illegible]  00
048  [illegible]  61
049  [illegible]  X:T
050  [illegible]  EXC
051  [illegible]  00
052  [illegible]  X:T
053  [illegible]  GE
054  [illegible]  00
055  [illegible]  61
056  [illegible]  RCL
057  [illegible]  00
058  [illegible]  X:T
059  [illegible]  STO
060  [illegible]  STO
061  [illegible]  0
062  [illegible]  X:T
063  [illegible]  0

Comps
064  76  LBL
065  42  STO
066  32  X:T
067  [illegible]  STO
068  [illegible]  01
069  [illegible]  X:T
070  [illegible]  STO
071  [illegible]  01
072  [illegible]  INT
073  [illegible]  STO
074  [illegible]  02
075  [illegible]  +/-
076  [illegible]  +
077  [illegible]  RCL
078  [illegible]  01
079  [illegible]  =
080  [illegible]  [illegible]
081  [illegible]  [illegible]
082  [illegible]  [illegible]
083  [illegible]  =
084  [illegible]  STO
085  [illegible]  [illegible]
086  [illegible]  RCL
087  [illegible]  [illegible]
088  [illegible]  [illegible]
089  [illegible]  -
090  [illegible]  EXC
091  [illegible]  00
092  [illegible]  +
093  [illegible]  -
094  [illegible]  [illegible]
095  [illegible]  [illegible]
096  [illegible]  [illegible]
097  95  =
098  42  STO
099  [illegible]  [illegible]
100  92  RTN
```

SUBROUTINES

SNS Analogs of Classical Operators

Unary ($\underset{\sim}{N}$)

```
101  76  LBL
102  75   -
103  22  INV
104  71  SBR
105  42  STO
106  [illegible]  [illegible]
107  [illegible]  [illegible]
108  85   +
109  43  RCL
110  00   00
111  65   X
112  93   .
113  01   1
114  95   =
115  92  RTN
```

Unary ($\underset{\sim}{Y}$)

```
116  76  LBL
117  35  1/X
118  22  INV
119  71  SBR
120  42  STO
121  43  RCL
122  00   00
123  32  X:T
124  43  RCL
125  01   01
126  71  SBR
127  44  SUM
128  85   +
129  43  RCL
130  01   01
131  65   X
132  93   .
133  01   1
134  95   =
135  92  RTN
```

Unary ($\underset{\sim}{Y} \equiv \overset{\leftarrow}{\underset{\sim}{Y}}$)

```
136  76  LBL
137  45  YX
138  32  X:T
139  71  SBR
140  42  STO
141  32  X:T
142  43  RCL
143  00   00
144  71  SBR
145  44  SUM
146  65   X
147  93   .
148  01   1
149  85   +
150  43  RCL
151  00   00
152  95   =
153  92  RTN
```

Binary ($\underset{\sim}{S}$)

```
154  76  LBL
155  95   =
156  71  SBR
157  79  [illegible]
158  43  RCL
159  00   00
160  65   X
161  43  RCL
162  02   02
163  95   =
164  32  X:T
165  43  RCL
166  01   01
167  65   X
168  43  RCL
169  03   03
170  95   =
171  71  SBR
172  44  SUM
173  42  STO
174  04   04
175  43  RCL
176  01   01
177  65   X
178  43  RCL
179  02   02
180  95   =
181  32  X:T
182  43  RCL
183  00   00
184  65   X
185  43  RCL
186  03   03
187  95   =
188  71  SBR
189  44  SUM
190  65   X
191  93   .
192  01   1
193  85   +
194  43  RCL
195  04   04
196  95   =
197  92  RTN
```

SNS Analogs of Classical Operators (Contd)

Binary (Q)

198	76	LBL
199	85	+
200	71	SBR
201	78	Σ+
202	43	RCL
203	00	00
204	32	X:T
205	43	RCL
206	02	02
207	71	SBR
208	44	SUM
209	42	STO
210	00	00
211	43	RCL
212	01	01
213	32	X:T
214	43	RCL
215	03	03
216	71	SBR
217	49	PRD
218	65	×
219	93	.
220	01	1
221	85	+
222	43	RCL
223	00	00
224	95	=
225	92	RTN

Binary (A)

226	76	LBL
227	65	×
228	71	SBR
229	78	Σ+
230	43	RCL
231	00	00
232	32	X:T
233	43	RCL
234	02	02
235	71	SBR
236	49	PRD
237	42	STO
238	00	00
239	43	RCL
240	01	01
241	32	X:T
242	43	RCL
243	03	03
244	71	SBR
245	44	SUM
246	65	×
247	93	.
248	01	1
249	55	÷
250	43	RCL
251	00	00
252	95	=
253	92	RTN

Purely SNS Operators

(Direct Operators)

With (W)

```
254   76  LBL
255   25  CLR
256   71  SBR
257   42  STO
258   43  RCL
259   00   00
260   32  X:T
261   43  RCL
262   02   02
263   71  SBR
264   49  PRD
265   42  STO
266   00   00
267   43  RCL
268   01   01
269   32  X:T
270   43  RCL
271   03   03
272   71  SBR
273   49  PRD
274   65   X
275   93   .
276   01   1
277   85   +
278   43  RCL
279   00   00
280   95   =
281   35  1/X
282   35  1/X
283   92  RTN
```

Upon (U)

```
284   76  LBL
285   55   ÷
286   71  SBR
287   42  STO
288   43  RCL
289   01   01
290   32  X:T
291   43  RCL
292   03   03
293   71  SBR
294   44  SUM
295   42  STO
296   01   01
297   43  RCL
298   00   00
299   32  X:T
300   43  RCL
301   02   02
302   71  SBR
303   44  SUM
304   85   +
305   43  RCL
306   01   01
307   65   X
308   93   .
309   01   1
310   95   =
311   92  RTN
```

Agree (C)

```
312   76  LB
313   34  ΓX
314   71  SB
315   42  ST
316   43  RC
317   00   0
318   32  X:T
319   43  RCL
320   02   02
321   71  SBR
322   52  EE
323   42  STO
324   00   00
325   43  RCL
326   01   01
327   32  X:T
328   43  RCL
329   03   03
330   71  SBR
331   52  EE
332   32  X:T
333   43  RCL
334   00   00
335   71  SBR
336   49  PRD
337   42  STO
338   00   00
339   71  SBR
340   68  NOP
341   65   X
342   93   .
343   01   1
344   85   +
345   43  RCL
346   00   00
347   95   =
348   92  RTN
```

Purely SNS Operators

Reverse Operators

Reverse or (Q)

```
349  76  LBL
350  22  INV
351  71  SBR
352  78  Σ+
353  43  RCL
354  00   00
355  32  X⇌T
356  43  RCL
357  02   02
358  71  SBR
359  49  PRD
360  42  STO
361  02   02
362  43  RCL
363  01   01
364  32  X⇌T
365  43  RCL
366  03   03
367  71  SBR
368  49  PRD
369  32  X⇌T
370  43  RCL
371  02   02
372  71  SBR
373  44  SUM
374  65   ×
375  93   .
376  01   1
377  85   +
378  43  RCL
379  00   00
380  95   =
381  35  1/X
382  35  1/X
383  92  RTN
```